SEPTEMBER 1985 VOL. 7 NO. 9

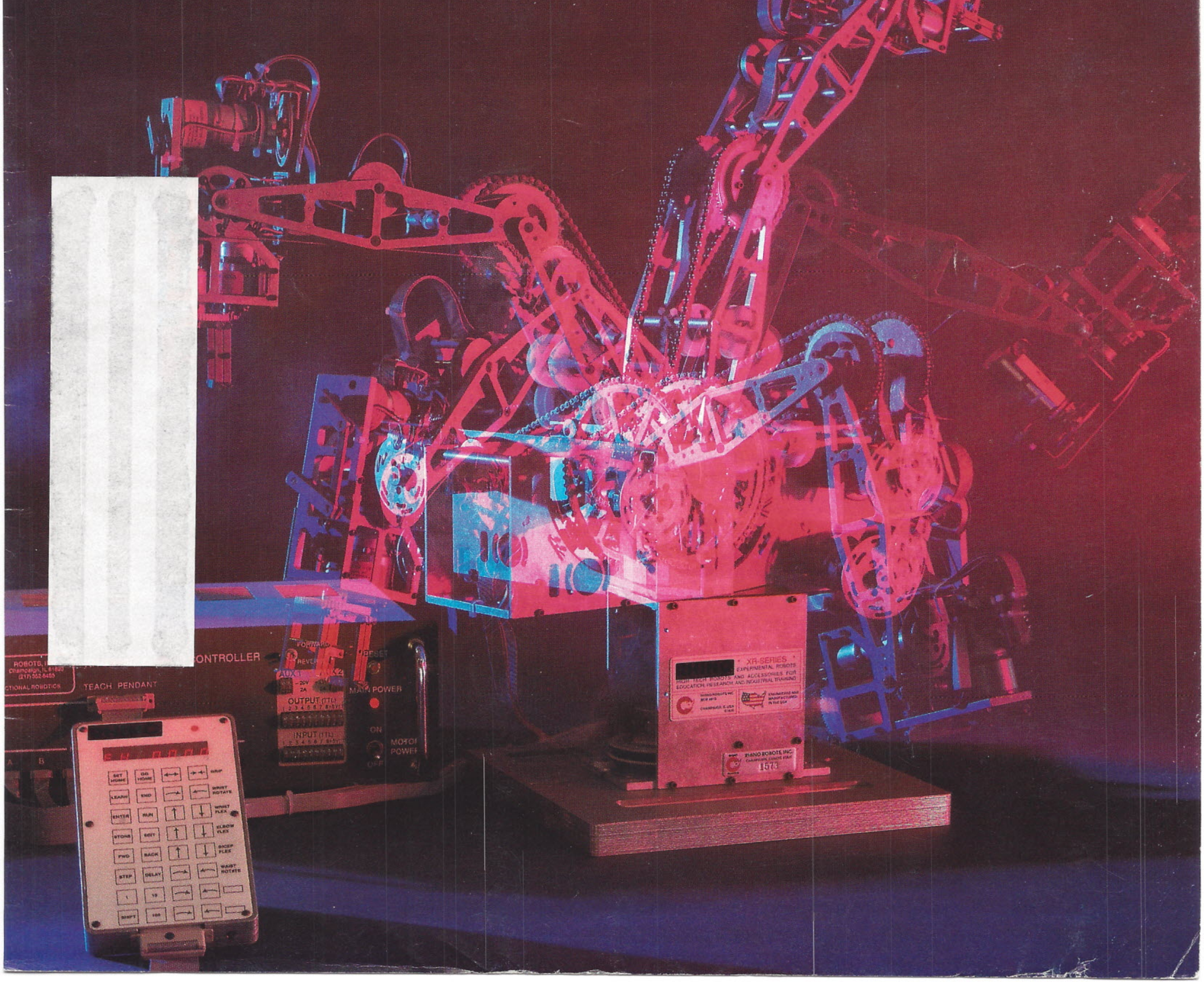
$3.00 USA/$3.50 CANADA

ROBOTICS AGE™

THE JOURNAL OF INTELLIGENT MACHINES

Robot Control Languages For Industry and Education

Robotics and Industrial Training: An Overview



ROBOTICS AGE—(ISSN 0197-1905) is published monthly by Robotics Age Inc., Strand Building, 174 Concord Street, Peterborough, NH 03458, phone (603) 924-7136. Address subscriptions, changes of address, USPC Form 3579, and fulfillment questions to *Robotics Age* Subscriptions, 174 Concord Street, Peterborough, NH 03458. Second class postage paid at Peterborough, NH and at additional mailing offices. **POSTMASTER:** Send address changes to ROBOTICS AGE, 174 Concord St., Peterborough, NH 03458.

Subscriptions are $24 for one year (12 issues), $45 for two years (24 issues), $63 for three years (36 issues) in the USA and its possessions. In Canada and Mexico, subscriptions are $28 for one year, $53 for two years, $75 for three years. For other countries, subscriptions are $32 for one year, surface delivery. Air delivery to selected areas at additional charges, rates upon request. Single copy price is $3 in the U.S., $3.50 in Canada and Mexico, $4 in Europe, and $4.50 elsewhere. Foreign subscriptions and single copy sales should be remitted in United States funds drawn on a U.S. bank.

Address all editorial correspondence to the Editor at *Robotics Age*, Strand Building, 174 Concord Street, Peterborough, NH 03458. Opinions expressed by the authors of articles are not necessarily those of *Robotics Age*. To aid in preparation of acceptable articles, an Authors' Guide is available upon request, if accompanied by a self-addressed 8½ by 11 inch envelope with first class postage for 3 ounces. Unacceptable manuscripts will be returned if accompanied by a self-addressed envelope with sufficient first class postage. Not responsible for lost manuscripts or photos.

ROBOTICS AGE™

THE JOURNAL OF INTELLIGENT MACHINES

SEPTEMBER 1985 VOL. 7 NO. 9

EDITORIAL

FEATURES

DEPARTMENTS

VIS IN REDUNDANTIA EST
Y.A.P.P.

About the cover: The Rhino XR-Series robot, shown here in multiple exposures, is a popular instructional device. A programming language, RoboTalk, has been created specifically for the Rhino (see page 15).

Editorial

Control Abstractions

BY CARL HELMERS

The year 1985 is a time for robotic languages, a time of creating new languages as well as of finding new uses for old ones. In this issue appear several articles on language themes. One discusses FORTH as a language to express algorithms in robotic applications. Another introduces a new interpretive robot language called RoboTalk. A third shows how a variant of LOGO can be used to investigate the problems of simulating mobile robots. These languages all represent different ways to express control abstractions, the computer programs that implement the flow of automated processes.

Among the earliest successful methods for robot control was the simple teach-by-example used with simple editing tools. In time, robot systems engineers were able to add interpretive languages as well as abstract expressions of traditional relay ladder diagram control expressions. Research laboratories have for some time now been tying FORTRAN, Assembly Language, FORTH, and other languages to real-world interfaces for robots.

Such languages and their implementations were a strong step up the abstraction ladder from literal representation toward symbolic expressions for actions. We are now seeing a further move forward in languages and tools for robotic application. One such innovation is Karel, a structured high-level language and engineering development environment from GMF Robotics. The name was chosen in honor of Karel Capek, who, in his 1920 play, *R.U.R.*, first introduced the word "robot." Karel's designers describe both the language and the software tool system in this month's *Robotics Age*. What follows here are our reactions to Karel as we have come to know it so far.

We had our first taste of Karel this June at the GMF booth at Robots IX and decided to learn more. At the end of June, the Karel team provided us with an advance copy of a users' manual and shortly thereafter Ken Stoddard and Mitchel Ward sent a draft of their article.

As a long-time collector of information about computer languages and their application environments, we were eager to dig deeper into this new language so we paid a visit to the GMF plant in Troy, Michigan on our way home from this year's National Computer Conference. GMF and its people were kind enough to give me a half day of hands-on familiarization with the language, its tool environment, its developers, and its technical capabilities.

At Robots IX we had observed Karel's resemblance to many block structured languages—Pascal and Ada in particular. In 20 years of working with computers we have at various times used FORTRAN, COBOL, VAL, PL/1, XPL, and Pascal as primary languages. We have also studied many other languages in engineering and editorial work, including Modula II, Ada, C, LISP, SmallTalk, LOGO, FORTH, APL, and BASIC. This language background and the obvious tool philosophy of the GMF robot/environment builders further kindled our enthusiasm. Karel is an interactive program development environment for use by manufacturing engineers, and not just another slow interpreter of a primitive BASIC-like language. In developing our argument, we shall consider some of the criticisms that have been or might be leveled against Karel.

SOME CRITICISMS OF KAREL

Every individual has preferences and every engineering application has specialized aspects. Both come into play in choosing a language for a particular purpose. Is Karel the "language to end all languages"? No. Neither is FORTH, Pascal, Ada, Modula II or any other language/environment combination. Karel is not any better, or any worse, than many modern computer languages used for specialized purposes. Much depends on the implementation. The GMF approach is not just a language, but a language plus a tool set. Many faults that will be found with "yet another robot language" will be based on a misunderstanding of its goals and aims.

Karel is clearly *not* a general purpose real-time control language like Ada, C, or Modula II, although such languages might be appropriate for *implementing* a Karel environment. These more powerful languages grow unwieldy relative to an intentionally limited language implementation such as Karel. However, for those interested in building applications, Karel will probably provide a more useful and integrated design environment, precisely because it is a limited tool specialized to a particular set of applications.

For example, the Karel language concept of a position vector—location plus orientation—would have to be grafted onto the other languages. Karel's strict separation of data and program, coupled with a "teach" mechanism *symbolically* linked to the program, is

an advanced level of abstraction that is neatly integrated into its environment. One could use a modified subset of Ada (DOD types may rise in protest), C, or Modula II to control similar systems. Such an augmented subset is not really the same language, however. If we are going to create a different language with extensions for special purposes, why not call it by a different name? One of the powers of modern high-level language technology is that it permits custom designing of variant languages for specific and limited applications. Each language, Karel included, can be designed with an application area in mind. This forms the philosophy of the development environment in which it will be placed.

Karel is not a graphic display and simulation language; its controller environment does not allow it to be. Nor is it necessarily the language in which one would express sophisticated pattern recognition and vision system algorithms. But it is intended to be the language which an applications engineer would employ to integrate a vision system into a Karel workcell by reference to appropriate libraries. These are more properly addressed by the designers of such systems using languages in which expression of these problems is more natural.

Karel's integration into an environment of robotic applications is a step up from the "monkey-see/monkey-do" approach of first-generation robot controllers. The expression of *algorithms* in symbolic form integrates naturally with the real-world data of movement, paths, sensor input, and interrupts. Karel has the abstract power to express point data, plus the ability to tie such abstract points to real-world locations, orientations, and information. Expression of global system variables of the workcell is a natural and symbolic part of Karel.

SOFTWARE MANAGEMENT PHILOSOPHY

Perhaps the most persuasive argument *for* using a block structured language such as Karel is a clear understanding of its environment. The following could be an argument for any one of a number of related languages such as Karel, Pascal, Ada, Modula II, PL/1—even C. The robots and robotic systems of the future are not going into isolated, stand-alone environments. They are going to be installed in a modern industrial organization in which the management problems of the real world are a major consideration. Structured languages—the "algol-like" family—are well adapted to managing large projects with real-world engineering personnel.

In the real world of projects, people come and people go. A well documented program is the only solution to the inevitable personnel turnover in engineering/programming staffs. A figure not unknown in computer technology is the "lone wolf," a creative programming genius who hardly communicates with his fellow creatures. A management that overly depends on an engineer who happens to be a lone wolf is headed for real trouble. What happens if the genius quits and takes all his mental documentation with him? The last thing a plant manager responsible for a robotics installation needs is a software engineering philosophy that completely bypasses complete and detailed documentation.

Languages that encourage experimentation *are* important to the development process and the quick, interactive test is the model sought by many designers. The problem comes in weighing interactiveness against the design documentation goal. Not all computer languages encourage documentation-oriented programming, but a well designed program in a modern structured programming language can be its own design document. When comments are used cleverly, any programming language can be made to produce eminently readable documentation of their functions. With block structured languages like Pascal and its progeny, including Karel, the documentation of the program *is* the program.

Before the modern era of powerful inexpensive computers there was a balancing act between the interactive needs of the design engineer and the practicalities of fitting the engineering effort into a large project. One of Karel's finer qualities is that it pays attention to a rational software management philosophy while still attending to the interactive needs of the designer/programmer of the application.

The Karel language, as implemented, is only one component of a tool environment. An advanced syntax-directed editor is also part of that environment. Yes, there is a compile-execute mode of operation in Karel that FORTH and Modula II programmers complain about. But the interactivity of program experiments is still there; the only compilation errors possible are the significant ones. Compilations in a Karel development environment will not fail for trivial syntax error reasons as would be the case for compilation of nearly every programming environment to date. And, given the fact that the Karel system is a one-user system based on a super fast 32-bit microprocessor in the lab, only an extremely large program will require more than a few seconds' delay between editing and running. We observed numerous quick and convenient program development/change cycles during our time at GMF.

Finally, there is that close tie between the language, its editor, its operating system, and its physical hardware environment. Much thought has gone into the Karel design as to the proper means of communicating with the global hardware entities of the real manufacturing environment. These hooks are symbolic. They are also generic. Instead of machine-specific motor-by-motor control of a robot and its joints, all computations and hardware directions are done in machine independent coordinates measured in standard engineering units. Thus, distance components (X,Y, and Z) of the Karel position vector are in millimeters and angles are in degrees. With the generic expression of algorithms, it is possible to make specific instances convenient to modify and debug using the teach pendant and its symbolic links to the program's source text.

IN CONCLUSION . . .

From what we have seen, Karel is an excellent software environment approach to the problem of applying industrial automation at the workcell level. As such, it is an integrated environment that provides the engineer with a set of easily understandable tools. Even if there were no further developments—such as completion of a stand-alone personal computer-based off-line version of the environment—Karel would be a highly desirable tool for practical application. ■

Calendar

SEPTEMBER

4–6. ORCAL '85 Expo. Anaheim Convention Center, Anaheim, CA. Contact: Public Relations Dept., Society of Manufacturing Engineers, One SME Dr., PO Box 930, Dearborn, MI 48121, telephone (313) 271-0777, or Public Relations Dept., American Society for Metals, Metals Park, OH 44073, telephone (216) 338-5151.

4–6. AIAA/NASA Symposium on Automation, Robotics and Advanced Computing for the National Space Program. J.W. Marriott Hotel, Washington, D.C. Contact: Louis P. Clark, Program Chairman, NASA Headquarters, Code DE, 600 Independence Ave., Washington, D.C. 20546, telephone (202) 453-1883, or Pamela Edwards, Program Administrator, AIAA Headquarters, 1633 Broadway, New York, NY 10019, telephone (212) 408-9778.

6–8. International Personal Robot Congress & Exposition. Moscone Center, San Francisco, CA. Contact: Sharon D. Smith, Chair, IPRC '85 Organizing Committee, 8822 S. Martin Lane, Conifer, CO 80433, telephone (303) 674-5650.

9. Semiconductor Equipment Communications Standard for Factory Automation. San Jose, CA. Contact: Jack Ghiselli, GW Associates, Inc., 645 Mills Ave., Los Altos, CA 94022, telephone (415) 948-2896. (To be repeated 20 September in Boston, MA.)

9–10. Second International Conference on Advanced Robotics. Keidanren Kaikan Bldg., Tokyo, Japan. Contact: Mr. A. Yasutake, Organizing Secretary, Japan Industrial Robot Association, Kikai Shinko Kaikan Bldg., 3-5-8, Shibakoen, Minato-ku, Tokyo, 105 Japan.

9–11. OEM Design '85. Philadelphia Civic Center, Philadelphia, PA. Contact: Penton Exposition Division, 122 E. 42nd St., New York, NY 10017, telephone (212) 867-9191.

10–12. Midcon/85. Chicago, IL. Contact: Nancy Hogan, Electronic Conventions Management, 8110 Airport Blvd., Los Angeles, CA 90045, telephone (213) 772-2965.

11–12. Mid-Atlantic Electronics Design and Production Exhibition and Conference '85. Valley Forge Convention and Exhibit Center, King of Prussia, PA. Contact: International Marketing Services Ltd., 1719 S. Clinton St., Chicago, IL 60616, telephone (312) 421-7000.

11–13. Fifteenth International Symposium on Industrial Robots. Keidanren Kaikan Bldg., Tokyo, Japan. Contact: Mr. Y. Komori, Organizing Secretary, 15th ISIR, Japan Industrial Robot Association, Kikai Shinko Kaikan Bldg., 3-5-8, Shibakoen, Minato-ku, Tokyo, 105, Japan.

16–19. Finishing '85 Conference and Exposition. Cobo Hall, Detroit, MI. Contact: Finishing '85, Association for Finishing Processes of Society of Manufacturing Engineers, One SME Dr., PO Box 930, Dearborn, MI 48121, telephone (313) 271-1080.

16–20. Knowledge Engineering Workshop. Expert Knowledge Systems, Inc., McLean, VA. Contact: M. James Naughton, Ph.D., President, Expert Knowledge Systems, Inc., 6313 Old Chesterbrook Rd., McLean, VA 22101, telephone (703) 734-6966.

17. Technology of Computer Integrated Manufacturing. Sunnyvale Hilton, Sunnyvale, CA. Contact: Rich Karlgaard, Warr, Foote & Rose, 111 Main St., Suite 3, PO Box 1290, Los Altos, CA 94023, telephone (415) 941-2820, or Dr. Robert W. Atherton, Sentry CIMS, Four Main St., Los Altos, CA 94022, telephone (415) 949-3860.

17–25. EMO Hannover. Hannover Fairgrounds, Hannover, Federal Republic of Germany. Contact: EMO General Commissariat, Messegelande, D-3000 Hannover 82, Federal Republic of Germany.

18–19. International Congress for Metalworking and Automation, Hannover, Federal Republic of Germany. Contact: EMO General Commissariat, Messegelande, D-3000 Hannover 82, Federal Republic of Germany.

18–20. Machine Vision in Electronics Manufacturing: A Hands-On Clinic. Omni Hotel, Baltimore, MD. Contact: Society of Manufacturing Engineers, Special Programs Dept., One SME Dr., PO Box 930, Dearborn, MI 48121, telephone (313) 271-1500. (To be repeated 23–25 September in Baltimore, MD.)

19–20. Microcomputers in Manufacturing. San Francisco, CA. Contact: Bob Stearn, The Yankee Group, 89 Broad St., 14th Floor, Boston, MA 02110, telephone (617) 542-0100.

20–21. The 7th Annual FORTH Convention. Hyatt Rickeys Hotel, Palo Alto, CA. Contact: FORTH Interest Group, PO Box 8231, San Jose, CA 95155, telephone (408) 277-0668.

23–24. EMCA Regional Seminar. Sunnyvale Hilton, Sunnyvale, CA. Contact: Electronic Motion Control Association, 230 N. Michigan Ave., Chicago, IL 60601, telephone (312) 372-9800.

23–24. Robots in Clean Room Applications. 25–26. Automated Electronic Assembly. Sheraton-Hartford Hotel, Hartford, CT. Contact: Diane M. Korona, Program Adminstrator, Robotics International of the Society of Manufacturing Engineers, One SME Dr., PO Box 930, Dearborn, MI 48121, telephone (313) 271-1500, ext. 390.

23–25. Space Tech '85. Disneyland Hotel Convention Center, Anaheim, CA. Contact: Gregg Balko, Technical Activities Dept., Society of Manufacturing Engineers, One SME Dr., PO Box 930, Dearborn, MI 48121, telephone (313) 271-1500, ext. 368.

23–27. Mathematical Modeling & Simulation. Sheraton National Hotel, Arlington, VA. Contact: Continuing Education Institute, 10889 Wilshire Blvd., Los Angeles, CA 90024, telephone (213) 824-9545. (To be repeated 4–8 November in Los Angeles, CA.)

24–25. Sensing Robots for the Automotive Industry Seminar. Detroit, MI. Contact: Robotic Industries Association, PO Box 1366, Dearborn, MI 48121, telephone (313) 271-7800.

30–4 October. Knowledge-Based Computer Vision. and Expert Systems: Applications in Robotics and Control. The Turing Institute, Glasgow, Scotland. Contact: George House, 36 N. Hanover St., Glasgow G1 2AD, Scotland, telephone (041) 552-6400.

OCTOBER

1–3. Dallas/Fort Worth Tool & Manufacturing Engineering Conference and Exposition. Infomart, Dallas, TX. Contact: Public Relations Dept., Society of Manufacturing Engineers, One SME Dr., PO Box 930, Dearborn, MI 48121, telephone (313) 271-1500, ext. 293.

2–4. Developing Careers: Issues for Engineers and Employers. Royal Sonesta Hotel, Boston, MA. Contact: William R. Anderson, Institute of Electrical and Electronics Engineers, Washington, DC office, 1111 19th St., N.W., Suite 608, Washington, DC 20036, telephone (202) 785-0017.

8–10. International Robot Conference and Exhibition. [illegible]delphia Civic Cent[illegible]ia, PA. Co[illegible] Conference Ma[illegible] Co., 331 W. Wesley St., Wheaton, IL 60187, telephone (312) 668-8100.

CANCELLED

Circle 7

Calendar

8–11. FASTEC '85 Conference. Georgia World Congress Center, Atlanta, GA. Contact: Patricia Jones, Society of Manufacturing Engineers, One SME Dr., PO Box 930, Dearborn, MI 48121, telephone (313) 271-1500, ext. 377.

9–11. Robots East. Bayside Exposition Center, Boston, MA. Contact: Jeff Burnstein, PR Manager, Robotic Industries Association, PO Box 1366, Dearborn, MI 48121, telephone (313) 271-7800.

15–17. Developing Robot Workcells. Norcross (Atlanta), GA. Contact: Diane Korona, Program Administrator, Society of Manufacturing Engineers Special Programs Dept., One SME Dr., PO Box 930, Dearborn, MI 48121, telephone (313) 271-1500, ext. 390.

15–17. The 4th International Conference on Flexible Manufacturing Systems. The 3rd International Conference on Automated Guided Vehicle Systems. The 6th International Conference on Automation in Warehousing. Alvsjo Trade Fair and Conference Centre, Stockholm, Sweden. Contact: IFS (Conferences) Ltd., c/o Stockholm Convention Bureau, Box 1617, S-111 86 Stockholm, Sweden, telephone (0) 8 23 09 90; or IFS (Conferences) Ltd., 35–39 High St., Kempston, Bedford MK427BT, England, telephone (0234) 853605.

17–18. National Conference and Exposition on Robotics and Automated Systems. Hilton Inn Bossier, Bossier City, LA. Contact: Dr. R. Michael Harnett, Meeting Chairman, Office of the Dean, College of Engineering, Louisiana Tech University, PO Box 10348 T.S., Ruston, LA 71272, telephone (318) 257-4647.

18–27. 1985 International Capital Goods Trade Fair. International Exposition and Trade Center, Cleveland, OH. Contact: Sandy Hensel, Director of Public Relations, 1985 International Capital Goods Trade Fair, 6200 Riverside Dr., Cleveland, OH 44135, telephone (216) 676-6000.

20–23. International Symposium on Laboratory Robotics '85. Boston Park Plaza Hotel, Boston, MA. Contact: Gerald L. Hawk, Ph.D., or Janet Strimaitis, Zymark Corp., Zymark Center, Hopkinton, MA 01748, telephone (617) 435-9501.

21–24. ISA/85 COMPUTEC. Philadelphia, PA. Contact: Fred E. Gore, Fisher Controls International, Inc., 8301 Cameron Rd., Austin, TX 78753, telephone (512) 834-7066.

22–23. Robot Justification Workshop. Chicago, IL. Contact: Robotic Industries Association, PO Box 1366, Dearborn, MI 48121, telephone (313) 271-7800.

22–24. SATECH '85: Systems & Applied Technology Conference & Exhibition. O'Hare Expo Center, Chicago, IL. Contact: SATECH '85, 2472 Eastman Ave., Bldg. 34, Ventura, CA 93003, telephone (805) 656-0933.

25–27. Forth Modification Laboratory (FORML). Stettenfels Castle, Frankfurt, Germany. Contact: Forth Interest Group, PO Box 8231, San Jose, CA 95155, telephone (408) 277-0668.

28–29. Automation Means Business. Hyatt Regency, Chicago, IL. Contact: Robotics Industry Service, Dataquest Incorporated, 1290 Ridder Park Dr., San Jose, CA 95131-2398, telephone (408) 971-9000.

NOVEMBER

4–7. AUTOFACT '85 Conference and Exposition. Cobo Hall, Detroit, MI. Contact: Tom Akas, Group Manager, Public Relations, Computer and Automated Systems Association/SME, One SME Dr., PO Box 930, Dearborn, MI 48121, telephone (313) 271-1500.

Circle 15

Calendar

NOVEMBER

6–8. Computer Integrated Manufacturing: Practical Applications in Your Plant. Washington, D.C. Contact: Cliff Hopkins, Continuing Engineering Education, the George Washington University, Washington, DC, 20052, telephone (800) 424-9773 or (202) 676-8521.

12–13. Robot Control Systems Workshop. Cincinnati, OH. Contact: Robotic Industries Association, PO Box 1366, Dearborn, MI 48121, telephone (313) 271-7800.

12–14. Chicago Tool & Manufacturing Conference & Exposition. O'Hare Exposition Center, Rosemont, IL. Contact: Public Relations Dept., Society of Manufacturing Engineers, One SME Dr., PO Box 930, Dearborn, MI 48121, telephone (313) 271-0777.

13–15. Robot Assembly in the Electronics Industry. Orientation into Machine Vision. Robot Safety. (One day each.) San Jose, CA. Contact: Robotic Industries Association, PO Box 1366, Dearborn, MI 48121, telephone (313) 271-7800.

13–15 and 18–20. Machine Vision in Automotive Manufacturing: A Hands-On Clinic. Ann Arbor, MI. Contact: Joanne Rogers, SME Special Programs Div., Society of Manufacturing Engineers, One SME Dr., PO Box 930, Dearborn, MI 48121, telephone (313) 271-1500, ext. 399.

20. R & D Limited Partnerships Seminar. Robot Export Seminar. Detroit, MI. Contact: Robotic Industries Association, PO Box 1366, Dearborn, MI 48121, telephone (313) 271-7800.

20–22. Expert Systems Short Course. Monterey, CA. Contact: Continuing Education Institute, 10889 Wilshire Blvd., Los Angeles, CA 90024, telephone (213) 824-9545. (To be repeated 4–6 December in Columbia, MD.)

Letter

Star Wars Defense Rebutted

I am always pleased to see issues of social concern addressed in technical magazines such as *Robotics Age.* Society requires the input of its technically competent citizens for survival in our age.

The May editorial entitled "Existence Proof . . ." makes some incorrect assertions regarding the President's Strategic Defense Initiative (SDI). The astounding technological requirements for a system that can destroy ICBMs in flight are well documented (see, for example, *Scientific American,* October 1984). Although the capability of destroying a missile with a beam of energy has been demonstrated, the vast majority of experts has concluded that at no time in the next several decades will nuclear weapons be rendered, as the President said, "impotent and obsolete." Let us assume these critics are wrong and that a 100 percent kill ratio may be obtained by the "Star Wars" system. There is still nothing to stop the Soviets from shifting to cruise missiles, bomber aircraft, and submarines for weapons delivery.

The possibility that nuclear weapons will be made obsolete by Star Wars is zero. A Star Wars system will never protect *all* our people, but it might protect *some* of our missiles. A greater degree of safety for the populace might be obtained with Star Wars only if the number of ICBMs on both sides is reduced. President Reagan is losing any chance for obtaining this reduction by insisting on proceeding with SDI. Without a reduction pact, SDI will serve only to destabilize relations with the Soviets.

I urge you to reexamine the technology and strategy underlying the Strategic Defense Initiative. There exists technology that should not be developed, and Star Wars is such a technology.

Scott A. Walter
PO Box 10973
Stanford, CA 94305

FORTH FOR ROBOT CONTROL

George Dooley
Real-Time Devices
PO Box 906
State College, PA 16804

Software is the critical element for any robotics application. Robots are inherently real-time devices and present the software developer with unique and often extraordinarily difficult problems. Robotics software differs from other applications in that nearly all robot control software is customized to specific industrial requirements—one robot might be spray painting a car or turning a bolt while another robot of similar design might be welding. Robotics software must take into account the widely disparate uses to which robots are put. No single piece of application software can effectively control every robot. A suitable robotics language must be adaptable to a variety of real-time applications.

The special characteristics a robotics programming language needs are these:

- Transportable. The programs generated by the language should be easily converted to run on a variety of computers.
- Readable/Modifiable. The program language should be easily read and understood so that it can easily be modified.
- Generalizable. The program should be easily adapted to new hardware.
- Interactive. The program should be easily and quickly tested with the hardware in actual use. (A running dialog between system and programmer occurs.)
- Debuggable. Errors should be easy to correct.
- Vertical. The program should be low-level and high-level at the same time.
- Extensible. The language can be made more powerful by adding to its architecture, allowing new structures to be "grafted" to itself.

Let us now survey some of the other candidate robotics languages. BASIC, although interactive and easy to learn, is unstructured, slow, and not easily extended. FORTRAN also is popular and fast but unstructured, not easily extended, and inflexible. The only means of extending BASIC and FORTRAN is by adding assembly language calls to the language. PASCAL and C, both structured languages, are not usually implemented interactively. Their typical implementation is stuck in an EDIT-COMPILE-LINK-DEBUG cycle that can be limiting to creativity and productivity. These languages are often unsuitable for robot control.

FORTH was invented in 1969 by Charles Moore to control radio telescopes. The language arose from his frustration with traditional languages as inflexible and unsuitable for control applications on a machine with limited resources. In developing FORTH, Moore created a novel and effective method of solving complex automation problems. Abandoning traditional languages with their rigidly fixed syntax, he created a stack-oriented language that is extensible and highly interactive.

FORTH is constructed around vocabularies. The root vocabulary is a collection of basic "words" or commands. Writing programs in FORTH consists of defining new words made up from sequences of already existing words. A vertical hierarchy can be developed, starting with low-level "primitives," and continuing toward more and more complex, abstract words.

Verticality is an important property of a robot control language. An effective language must be a compromise between the computer that understands only binary logic levels and the user who prefers to communicate in his or her native language. A computer language should show the same flexibility as a human language; when a new situation or process arises, new words can be created to describe and control it. Automation language must encourage the creation of new language extensions that appear to the user as human language but remain fast and efficient in implementation.

The vertical hierarchy allows the user to link together language primitives to create new commands specific to the application. For use in robotics, the programmer can create commands to control various parts of the robot. These commands can, in turn, be used to create other words of increasing power, complexity, and abstraction. Thus, programming in FORTH consists of extending the root language to create another language and solving the problem in this new language. FORTH has accurately been described as a "meta-language," a language for writing application-oriented languages. As such, its character is entirely different from the formal flavor of conventional computer languages.

For example, to sample an analog-to-digital converter we might create a word called ADC that would take care of beginning the analog conversion, testing for completion, and reading the results. All the programmer has to do is enter ADC and the result is left on the stack. The word ADC actually becomes part of the language and can be used to create additional commands. Any word could have been chosen to name this process, but "ADC" is convenient for mnemonic recognition.

A structured language is desirable for robotics applications so that a project can be implemented in the correct top-down

manner. Adherence to the principles of structured programming makes a program easy to understand, debug, and change. BASIC and FORTRAN, being unstructured, are not suited for producing and maintaining large, complex control programs. Structured programming promotes careful program design and provides a means of controlling the runaway complexity of software. Unstructured programming results in programs of "spaghetti code." Such code is difficult to maintain and even to comprehend. A well designed structured program can readily accommodate changes in computer hardware. Languages like Pascal (and, to some extent, C) carry the structured programming idea of data and program constructs to a logical conclusion—at the expense of interactiveness.

An interactive language is important for the rapid software development that keeps programming costs down. A language that provides the necessary tools to write software rapidly permits alternative routes to be programmed and explored to create the best software model. Charles Moore refers to this aspect of FORTH as "serendipitous" in that the language lends itself to accidental discoveries. It is so easy to use that programmers often find themselves writing a small program or just playing around to see what happens. Often what happens is something significant that sheds light on the process and allows a deeper understanding of it.

The importance of an interactive language cannot be overemphasized. BASIC is burned permanently into ROM in millions of home computers, not because it is particularly powerful, elegant, or structured but because it is interactive. Human beings love to see the results of their work immediately. FORTH users often draw an analogy between programming and painting a picture. Imagine the frustration a painter would feel having to wait 10 to 15 minutes to see the effect of each single brushstroke. This is exactly the situation programmers face when they use a traditional high-level language in their conventional implementation. An editor is used to enter source code, the source must be compiled and linked, and finally executed to detect any bugs. The process is laborious, time-consuming, and required for each program change. This time delay hinders the creative process, breeds frustration, and discourages experimentation. FORTH breaks out of this EDIT-COMPILE-LINK-DEBUG cycle because its compiler and editor are FORTH words and it does not need a linker. A FORTH word can be defined interactively and tested exhaustively before it is incorporated into a higher-level definition. FORTH's approach to productivity is to break down the traditional barriers between language, utilities, and operating system.

The stack orientation of FORTH is its most often criticized characteristic: "FORTH would be such a great language if it weren't for the reverse Polish nonsense." In fact, the reverse Polish stack orientation is a powerful and fast method of passing arguments between words. In traditional languages, subroutines are a useful means of executing a frequently used piece of code. A heavy speed penalty is paid for the use of subroutines in other high-level languages since each parameter is passed to the subroutine through an indirect and complicated route and each argument returned is passed through this same circuitous process. FORTH's words act much like subroutines, except that instead of being called they need only be invoked. FORTH words expect or leave their results on the stack and the speed penalty is small.

Conventional wisdom says that all interactive languages are slow. Yet, FORTH executes as fast as, or faster than other high-level languages. It also provides full access to the computer's hardware. FORTH is as comfortable twiddling the bits at some controller port or reading a robot's sensor as it is creating powerful high-level commands. FORTH is fast because it compiles into a sequence of calls to the root kernel, which is a set of machine language routines.

Many commercial versions of FORTH have descended from the public domain FORTH Interest Group (FIG) version. FIG FORTH is useful for introducing the language to the public and is suitable for smaller control applications. For more sophisticated applications, PolyFORTH from FORTH, Inc. may be more useful. PolyFORTH is a professional system geared toward high-performance applications and includes some features that FIG descendants lack. For example, PolyFORTH has TICKS, a 32-bit variable that contains the number of milliseconds since midnight, an invaluable piece of information for real-time control. PolyFORTH also provides a turnkey compiler (to generate stand-alone applications), sealed vocabularies, and multiuser capabilities. Unlike many other commercial versions, PolyFORTH supports a number of background tasks. Multitasking is important for robots since they must often perform their tasks, monitor other situations, and manage communications in real time.

A new development is the recent release of the NC4000A single chip FORTH engine from Novix, Inc., 10590 N. Tantau Ave., Cupertino, CA 95014. Designed by Moore, the chip directly executes high-level FORTH at 10 million operations per second, faster than some mainframe computers. Such capability in a single chip suggests that FORTH will not only survive, but will have robotics application for some time to come.

George Dooley is vice president of Real-Time Devices.

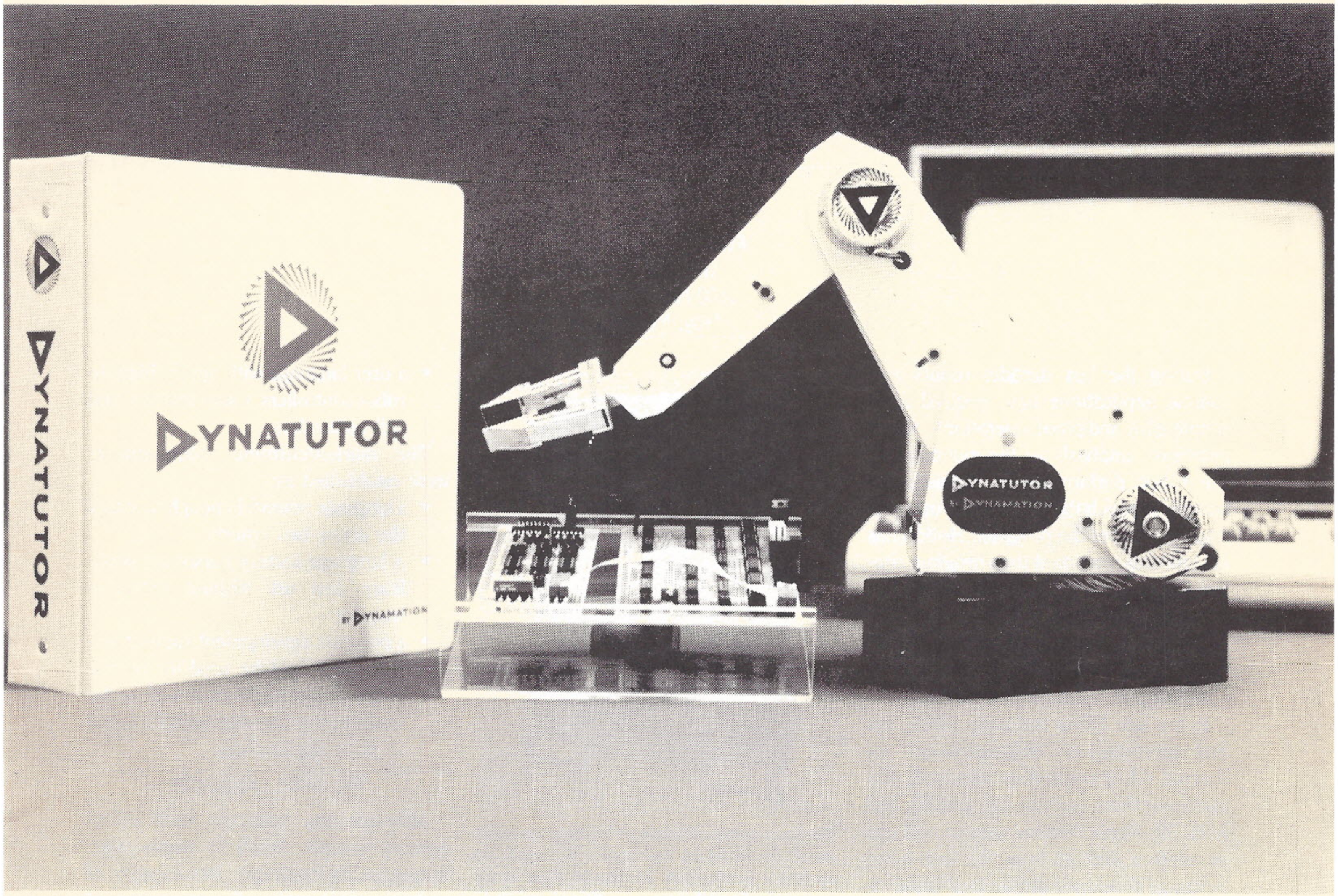
Reader Feedback

To rate this article, circle the appropriate number on the Reader Service card.

70	80	90
Excellent	Good	Fair

DYNATUTOR

A LOW COST EDUCATIONAL COMPUTER INTERFACING AND ROBOTICS PACKAGE



DYNATUTOR COMES COMPLETE WITH:

- FULLY ASSEMBLED ROBOT
- INTERFACE CIRCUIT
- EDUCATIONAL MANUAL
- SOFTWARE*
- POWER SUPPLY
- CABLES*

DISTRIBUTOR ENQUIRIES WELCOME

DYNAMATION

P.O. BOX 231, STATION "U", TORONTO, ONTARIO M8Z 5P1 • TELEPHONE (416) 927-0800

Circle 9

* *For the popular microcomputers.*

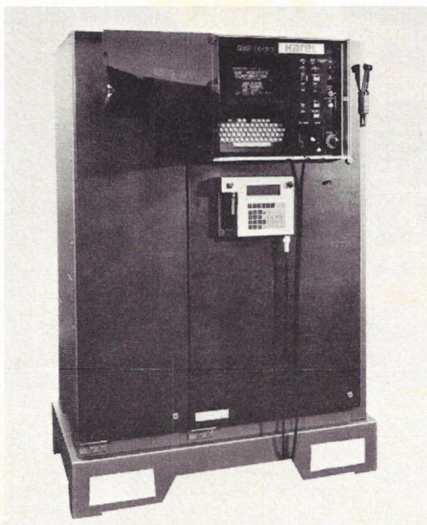
KAREL: A PROGRAMMING LANGUAGE FOR THE FACTORY FLOOR

Mitchel R. Ward
and
Kenneth A. Stoddard
GMF Robotics
5600 New King St.
Troy, MI 48098

During the last decade, robots and robotic applications have evolved from simple pick-and-place operations, which put heavy emphasis on the motions that the robot performed, to sophisticated systems in which the motions are only one small portion of the program. Motion has become less important than program control logic, interfacing, communications, operator interaction, and program computation. It is not uncommon today for 80 to 95 percent of a robot program to be composed of nonmotion constructs.

In the late 1970s and early 1980s, the first generation of "language-based" industrial robot controllers appeared. Examples of these are VAL [1], Rail [2], and AML [3]. These systems were in one way or another derived from the first robot language developed at Stanford University, WAVE [4], and its successor AL [5]. First-generation languages had simple control structures (IF GOTO), integer arithmetic, limited coordinate transformations, limited user interrupt handling, keyboard/CRT input/output (I/O), and relatively slow interpretive execution.

In 1983, the first of the second generation of robot languages, VAL-II [6], was introduced. Second-generation languages are characterized by complete control structures, varied data types with full arithmetic support, powerful user interrupt handling with process control support, full I/O including teach pendant and secondary storage, and faster user program execution.



The Karel control cabinet door has been opened to reveal the keyboard and screen (left) and the operator's panel (right). On the far right of the cabinet is a fused flange power disconnect switch. The teach pendant hangs on the front of the cabinet below the keyboard.

Karel is a new second-generation robot language recently created and implemented at GMF Robotics [7]. Karel is also supported on GMF vision and off-line programming products and will be the basis of GMF cell control products.

DESIGN PHILOSOPHY

The design philosophy for Karel was to develop:

- a user programming system to support special-purpose application software development on top of (not within) the basic controller
- a user language with applicability to robot controllers, vision systems, and cell controllers

The market/customer requirements were established as:

- a language powerful enough to satisfy the above two criteria
- a language factory personnel could learn and use without extensive training
- a program development support system that could be used for off-line program development as well as the on-line programming typically found in plant applications

THE KAREL LANGUAGE

Although the language itself incorporates features found in many other languages, its integration into a system of software tools makes the programming and operation of a Karel language program unique. Karel is a high-level programming language that combines certain features of Pascal, some characteristics of first-generation robot programming languages, and finally some extensions unique to Karel. The structure and syntax of Karel are designed to provide a language that is easy to learn for simple applications while providing the sophistication and power necessary for advanced applications involving sensors, communications, a high degree of operator interaction, and extensive process control.

Simple Data Types. Karel provides four basic scalar data types: integer, real, string,

ROBOTICS AGE BACK ISSUES

SUMMER 1979: Digital Speed Control of DC Motors; Industrial Robots 79; Introduction to Robot Vision; The Grivet Chess-Playing Arm. **(photocopy)**

WINTER 1979: Advances in Switched-Mode Power Conversion, Part 1; Prospects for Robots in Space; Robotics Research in Japan, Report from IJCA16.

SPRING 1980: Microcomputer Based Path Control; Robotics Research in France; Multiple Sensors for a Low-Cost Robot; the Robots of Autofact II; Inside Big Trak.

SUMMER 1980: Industrial Robots; Today and Tomorrow; Introducing the Minimover 5; Advances in Switched Mode Power, Part II.

FALL 1980: Using Optical Shaft Encoders; Interview with Victor Scheinman; Robot Vision for Industry; The Autovision System; Industrial Robotics '80; Robots on Your Own Time; Superkim Meets ET-2.

MAR/APR 1981: Video Signal Input; Chain-Code; Camera Geometry for Robot Vision; TIG Welding with Robots; Robot Digestive Track—Robots on Your Own Time.

JULY/AUG 1981: Segmenting Binary Images; The Robot as Transfer Device; Continuous Path Control of Stepper Motors; TIMEL: A Homebuilt Robot.

SEP/OCT 1981: Bullish Days in the Robot Business: Edge Detection in Man & Machine; Continuous Path Control with Stepping Motors; Build a Low-Cost Image Digitizer, Report from JACC-81; The Robot Builder's Bookshelf.

NOV/DEC 1981: Teach Your Robot to Speak; Fast Trig Functions for Robot Control; An Interview with George Devol; The Great Japanese Robot Show; TIMEL: A Homebuilt Robot, Part II.

JAN/FEB 1982: Avatar: A Homebuilt Robot; A Look at SS-50 Computer Boards; Working Within Limits; Ambulatron: Another Contest Winner; Quester.

MAR/APR 1982: The Rhino XR-1: A Hands-On Introduction to Robotics; Power for Robots; A Computer Controlled Sentry Robot: A Homebuilt Project Report; Natural Language Understanding: A First Look; RT-13 Video/Sound Recognition System; An Inexpensive Hand; Type 'N Talk.

MAY/JUNE 1982: Part Sources for Robots; An Inexpensive Arm-Hand System; The Polaroid P100 Polapulse Battery: Solution Waiting for a Problem; New Robot Books for the Bookcase: Applying Robot Vision to the Real World; Robots VI: A Landmark in an Exciting Era; Photo Essay and Notes from Robot VI.

JULY/AUG 1982: The Microbot Teach-mover; Some Notes On the Rhino XR-1 and Minimover 5; Patent Probe; Use Your Apple As a Robotics Development System; IBM Robots; Adapting a Speech Synthesizer; Constructing an Intelligent Mobile Platform, Part I.

SEPT/OCT 1982: Roving Robots; Report on SIGGRAPH '82; Patent Probe No. 4,221,997; Constructing an Intelligent Mobile Platform, Part II; The Physics of One-Legged Mobile Robots.

NOV/DEC 1982: Robot Wrist Actuators; Patent Probe; A Microcomputer Based, Real-Time Robot System; The Physics of One-Legged Mobile Robots; Part II; 1982 AAAI Conference; Armatron: A Study in Arm Engineering; Invention Documentation: A Primer.

JAN/FEB 1983: The Move-Master RM-101; Mailmobiles in the Office; Teaching the Rhino XR-1 to Write; The Philosophy and Birth of Computer Science; The 2-Roll Gripper.

MAR/APR 1983: Nuclear Power Plant Emergency Damage Control Robot; Artificial Intelligence and the Nature of Robotics; Patent Probe: Driverless Vehicle Autoguide; Lamberton Robots.

MAY/JUNE 1983: Patent Probe: Multi-Purpose Mechanical Hand; XY Interpolation Algorithms; Designing With Optical Shaft Encoders; A Table of Contemporary Manipulator Devices; An Algorithmic Approach to Intelligent Robot Mobility.

JULY/AUG 1983: The Get Away Special, Part I; A Nose for the Heath Hero-1; Patent Probe: Ambulatory Platform: Robotics and the Law: Organizing the Venture; A Table of Contemporary Manipulator Devices.

SEPT/OCT 1983: ODEX 1: The First Functionoid; A Cog-Wheel Driven Robot Cart; A Table of Contemporary Manipulator Devices; Patent Probe: A Portable Robot Task Analyzer; The Penpad: Handwritten Input for Computers; The Get Away Special, Part II: Flight Preparations.

NOV/DEC 1983: Complete Control with Forth on a Chip; A Simple Sense of Touch for Robotic Fingers; Current Offerings in Robotics Education; Single Board Computer Manufacturers; A Table of Contemporary Manipulator Devices; Welding Apparatus with Vision Correction.

JANUARY 1984: Robots in Batch Manufacturing; Super Armatron; Directory of Robotics Education and Training Institutions; A Table of Contemporary Manipulator Devices; Patent Probe: A New Robot Patent Category; Operator Roles in Robotics; The Scorpion: Software Overview.

FEBRUARY 1984: GRASP: From Computer Aided Robot Design to Off-Line Programming; Design and Construction of a Five-Fingered Robotic Hand; Patent Probe: Omnidirectional Vehicle; Using Microprocessors with Radio-Control Servos; Designing a Reliable Voice-Input Robot Control Language; Part II, The Scorpion: Motor Control Instructions.

MARCH 1984: Part I, Armega 33: Mechanical Design; Patent Probe: Robot Computer Chess; Part III, The Scorpion: Commands with Responses; A Simple Computer Interface.

APRIL 1984: Closing the Sensor-Computer-Control Loop; Bipedal Balance; Patent Probe: Robot Warehouse; Part II, Armega 33: The Electrical Components.

MAY 1984: Backyard Foundry for Sand Casting; Androtext, A High-Level Language for Personal Robots; Part III; Armega 33: Computer Control; Computers in the Real World.

JUNE 1984: Introduction to Numerical Control Programming; Glossary of CNC and Machine Tool Technology; A Glance at Some Microprocessor-Controlled CNC Tools; Intel's Bitbus Microcontroller Interconnect.

JULY 1984: Shape Memory Effect Alloys for Robotic Devices; Hitachi's Robot Hand; Part II, Backyard Foundry for Sand Casting: Making the Molds; Fantasy Meets Reality at New York Robot Exhibit; Meccano and the Home-Built Robot; The Role of Robots in Flexible Manufacturing Systems; A Review of the Colne Armdroid I.

AUGUST 1984: Animate vs. Inanimate; The Echo II Speech Synthesizer; Computer Vision and Ranging Systems for a Ping Pong Playing Robot; Robots and Employment; Angles and Brads; Robots 8 Exposition and Conference; Part of the Beginning.

SEPTEMBER 1984: Tabletop Technology; Robots in the Classroom; Robots in Elementary Education; Moravec's Mobile Robots; Robotics' Role in HighTech Education; Using the Polaroid Ultrasonic Ranging System.

OCTOBER 1984: Introduction to Data Acquisition; A Simple Thermal Touch Sensor; Experimenting Made Easy; The ADC-1 Data Acquisition System; Of Mice and Machines.

NOVEMBER 1984: Part I, A Lisp-Based Robot Control System; Part I, Introduction to Prolog; EARL: An Easy Robot Language; Product Notes: The String Computer; A Biological Model for Multilegged Robot Control.

DECEMBER 1984: Motion Control for Automation and Robotics; A Lisp-Based Robot Control System, Part II; Robotix, Part I; Introduction to Prolog, Part II; A Third- Generation Stepper Motor Controller, Part 1.

JANUARY 1985: The Walking Gyro; Nomad MK III: The Flight of Fancy; Robotix, Part II: External Control; A Lisp-Based Robot Control System, Part III: An Example Environment; Rethinking Robot Mobility; A Pneumatic Walking Robot.

FEBRUARY 1985: The Sandi Project; Photo Essay: A First Glimpse at Gemini; Robotics Research: The Next Five Years and Beyond; Report from the Unmanned Vehicles Conference; A Modular Remote Data Acquisition System Design; Keeping the HERO I's Steering Wheel Straight; A Third-Generation Stepper Motor Controller: Part II.

MARCH 1985: Miniature CCD Cameras: A New Technology for Machine Vision; Vision and the Computer: An Overview; PC-Based Image Processing; Polaroid Ultrasonic Ranging Sensors in Robotics Applications.

APRIL 1985: Speech Recognition: Machines that Listen; Interview with Joseph Engelberger; The Listeners: Intelligent Machines with Voice Technology; Fourth for Computer Vision in Industrial Applications; A Second-Generation Autonomous Sentry Robot.

MAY 1985: The ABCs of X-Y Positioning; A Compliant Mechanical Gripper; Third-Generation Robots: Their Definition, Characteristics, and Applications; A Critique of Three Gripper Designs; Tool-Changing Robot Hands.

JUNE 1985: Justifying the Robotics Replacement Decision; A New Computerized Tomographic-Aided Robotic Stereotaxis System; A Z80-Based, Multitasking Control System; Recognizing Parts That Touch.

JULY 1985: A New Super Camera for Vision Systems; Force Sensors for Robotic Assembly Systems; A Mutielement Ultrasonic Ranging Array; Goal-Seeking Sensor; Robots and Safety: An Industry Overview.

AUGUST 1985: The Exact Inverse Kinematics Solutions for the Rhino XR-2 Robot Manipulator; Previewing Robotic Motion with Computer Graphics; Inverse Robot Kinematics Derived from Planes of Movement; Graphics Simulation and Programming for Robotic Workcell Design.

______	Copies Summer '79 (photocopy)	Vol.1 No.1	$15/ea.
______	Copies Winter '79	Vol.1 No.2	$3/ea.
______	Copies Spring '80	Vol.2 No.1	$3/ea.
______	Copies Summer '80	Vol.2 No.2	$3/ea.
______	Copies Fall '80	Vol.2 No.3	$3/ea.
______	Copies Mar/Apr '81	Vol.3 No.2	$3/ea.
______	Copies July/Aug '81	Vol.3 No.4	$3/ea.
______	Copies Sept/Oct '81	Vol.3 No.5	$3/ea.
______	Copies Nov/Dec '81	Vol.3 No.6	$3/ea.
______	Copies Jan/Feb '82	Vol.4 No.1	$3/ea.
______	Copies Mar/Apr '82	Vol.4 No.2	$3/ea.
______	Copies May/June '82	Vol.4 No.3	$3/ea.
______	Copies Jul/Aug '82	Vol.4 No.4	$3/ea.
______	Copies Sept/Oct '82	Vol.4 No.5	$3/ea.
______	Copies Nov/Dec '82	Vol.4 No.6	$3/ea.
______	Copies Jan/Feb '83	Vol.5 No.1	$3/ea.
______	Copies Mar/Apr '83	Vol.5 No.2	$3/ea.
______	Copies May/June '83	Vol.5 No.3	$3/ea.
______	Copies July/Aug '83	Vol.5 No.4	$3/ea.
______	Copies Sept/Oct '83	Vol.5 No.5	$3/ea.
______	Copies Nov/Dec '83	Vol.5 No.6	$3/ea.
______	Copies January '84	Vol.6 No.1	$3/ea.
______	Copies February '84	Vol.6 No.2	$3/ea.
______	Copies March '84	Vol.6 No.3	$3/ea.
______	Copies April '84	Vol.6 No.4	$3/ea.
______	Copies May '84	Vol.6 No.5	$3/ea.
______	Copies June '84	Vol.6 No.6	$3/ea.
______	Copies July '84	Vol.6 No.7	$3/ea.
______	Copies August '84	Vol.6 No. 8	$3/ea.
______	Copies September '84	Vol.6 No. 9	$3/ea.
______	Copies October '84	Vol.6 No.10	$3/ea.
______	Copies November '84	Vol.6 No.11	$3/ea.
______	Copies December '84	Vol.6 No.12	$3/ea.
______	Copies January '85	Vol.7 No.1	$3/ea.
______	Copies February '85	Vol.7 No.2	$3/ea.
______	Copies March '85	Vol.7 No.3	$3/ea.
______	Copies April '85	Vol.7 No.4	$3/ea.
______	Copies May '85	Vol.7 No.5	$3/ea.
______	Copies June '85	Vol.7 No.6	$3/ea.
______	Copies July '85	Vol.7 No.7	$3/ea.
______	Copies August '85	Vol.7 No.8	$3/ea.

9/85 RA

Send to:

ROBOTICS AGE
174 Concord St.
Peterborough, NH 03458

VISA MasterCard

☐ ***SPECIAL OFFER:***

With my order of 10 or more back issues, please send me FREE, the photocopy of the Summer 1979 issue—a $15.00 value!

Send in your name, address, and payment to receive any back issues of ROBOTICS AGE you've missed. Payment must accompany order—back issues *cannot* be billed.

For individual back issues add $1.50 postage and handling for the first back issue. For each additional back issue add 50¢.

Cost of Back issues ______________

Postage and Handling ______________

Total Payment Enclosed ______________

NAME ______________

ADDRESS ______________

CITY ______________ STATE ______________ ZIP ______________

and Boolean. These types are available as both constant and variable data. Integer data includes whole numbers in the range ± 2,147,483,647. Real data is supported as 32-bit floating point data. String data is strings of any printable characters. Boolean data has two possible values, TRUE (ON) or FALSE (OFF). One-dimensional arrays of scalar data types are supported in Karel. Arrays must be declared explicitly with a length (number of elements).

Structured Data. In addition to scalar data types and arrays of scalar data, Karel supports structured data. Structured data is distinguished from scalar data in that it consists of a set of values that have a more complex form than a simple array of scalars. Examples are position data, vectors, auxpos (auxiliary axis position data), and paths.

Positional data is used to define location and orientation. It might refer, for example, to variable information such as the location and orientation of an object or information such as the Tool Center Point (TCP). In Karel, positional data is represented as a location (X,Y,Z) and an orientation (W,P,R) in a Cartesián space.

Paths are arrays of positions with each path node being one position. A feature of paths is that, in addition to position data, other data such as process data may be associated with a node of a path. Nodes can be arbitrarily added to or deleted from a path during teaching. Paths provide a means of easily specifying a sequence of positions through which the robot is to move, and to specify the motion with a single statement. Paths are convenient for continuous path applications such as painting, sealing, and arc welding, and for specifying motions around obstacles where the number of required taught points is not known.

Vectors are composed of three elements. The numeric values of these elements are similar to the 3-component vectors of mathematics. They are typically used to construct position values from within a Karel program, or for other vector functions and quantities.

Karel supports auxiliary axes for such functions as positioner control for arc welding. AUXPOS is the data type used to record the positional values for these additional axes. Special motion statements are provided for specifying the motion of these axes and for coordinating their motion with the robot axes. Jogging the auxiliary axes and teaching auxiliary axis position are supported by the teach pendant.

Expressions. Mathematical and relational operators are defined for all data types. The basic arithmetic operators are addition (+), subtraction (-), multiplication (*), real division (/), and integer division (DIV). Trigonometric functions are also supported. The relational operators include equal (=), not equal (<>), greater than (>), less than (<), greater than or equal to (>=), and less than or equal to (<=). The logical operators are OR, AND, and NOT. A full set of mathematical operators and built-in routines is provided to support structured data types. This set consists of vector addition (+), scaling (*), dot product (@), cross product (#), and coordinate transformations (:).

Motion Statements. Motion statements can specify motion of the robot TCP as well as that of auxiliary axes. Karel provides a broad selection of basic motion statements (MOVEs). They include:

```
MOVE TO        moves to a specified location
MOVE NEAR      moves near a position along TOOL Z axis
MOVE AWAY      moves away from the current position along
               TOOL Z axis
MOVE AXIS      moves an individual axis
MOVE RELATIVE  moves the robot relative to its current
               position
MOVE ABOUT     rotates around a specified vector
MOVE ALONG     moves along a path
```

A number of special motion control and monitoring clauses can be added to individual motion statements to temporarily change the type of a motion or to specify conditions that should be monitored throughout the motion. A NOWAIT clause on a MOVE statement tells the program interpreter that as soon as the specified motion is initiated, the interpreter can proceed with executing subsequent statements.

Textual I/O. User-programmed textual (character, string) I/O is provided to the computer terminal, the teach pendant, and to bubble memory files. The Karel statements READ and WRITE provide both formatted and unformatted I/O operations. This feature permits application programs to be designed and implemented with a high degree of operational information flow. Meaningful operational, error, prompting, and status messages can be provided during daily operation to inform the operator of the operational status and to specify required actions.

Process Control I/O. The basic process control support is via digital and analog I/O. Karel systems in a normal hardware/software configuration support up to 128 separate digital I/O points. Optional configurations support additional I/O. Digital I/O can be referenced as individual lines or as a group and treated as binary values. Individual lines can be turned ON, OFF, or PULSED. Analog I/O is also provided for welding controllers, sensory inputs, and general process control. The program constructs for performing this I/O use reserved symbols to represent each type of I/O. Assignment statements are used to set the values of individual I/O points:

```
DOUT[3] = ON   turns on digital output 3
AOUT[2] = 346  sets second analog output to 346
IF DIN[5] ...  tests input 5 to be ON
GPOUT[3] = 10  sets value 10 on digital output lines defined
               for group 3
```

Logic Control. Karel provides all the constructs of a high-level programming language. The FOR, REPEAT, and WHILE statements provide the basic looping constructs. The IF-THEN-ELSE, and SELECT (case) provide the standard alternative selection statements. The GOTO statement is provided to handle the exceptional situations when the five basic constructs are not adequate.

Monitors and Interrupts. Karel provides two types of user monitors and interrupts. Local interrupt handlers are provided that are enabled for only a single robot statement such as:

```
MOVE TO pickup UNTIL DIN[1] = ON
```

This statement moves the robot TCP to the position defined by "pickup." Normally, the motion would terminate when the robot TCP reached pickup. The UNTIL clause specifies that the motion is terminated when either the robot reaches pickup or when digital input 1 comes on. Program execution then continues at the next statement.

Global monitors specify a condition to be monitored globally during program execution. The action to be taken when the condition is satisfied is specified by a WHEN statement. Global monitors can be enabled and disabled as needed by the application program.

```
WHEN [1] DIN [Safety_gate] DO SHUTDOWN
```

This example establishes a monitor for a digital input, asking for an interrupt service routine (SHUTDOWN) to be executed

when the gate is opened. The difference between a service routine and all other routines is that parameters cannot be enabled and disabled as needed by the application program.

WHEN clause. They apply only to a single statement. In the following example, when the robot is 15 ms from POSITION [1], the digital output is turned on.

```
MOVE ALONG PATH 1,
  WHEN TIME 15 BEFORE POSITION [1]
    DO DOUT[GRIPPER] = ON
```

Routines. Karel supports both subroutines and functions that return values, including parameters, local data, and nesting to an arbitrary depth. Routines are invoked simply by using the name of the routine.

Importing Data and Routines. The sharing of data between application programs is an important feature for creating special application packages and for setting up data to be shared among programs either on the same robot or across all the robots in a given application area. In Karel, program data created for or by one program can be accessed by another program. Key reference positions for a particular application can be taught once and shared by several programs without maintaining multiple copies. One example is an operator setup procedure for vision calibration. Using shared data, a program can be written that leads the operator through the calibration and creates the positional data required for daily operation. This data can be loaded later and used with the actual operational program.

PROGRAM DEVELOPMENT

Karel, as implemented at GMF, is part of a full set of software development tools specialized for robotic, vision, and cell controller projects. This environment also allows off-line program development on computers separate from the actual robot systems.

Program Creation. Karel programs are created using a sophisticated line-oriented editor that knows the Karel language syntax. This essential tool provides commands for program entry and insertion, deletion, and replacement of text. During text entry the editor can optionally check for syntax errors within a program. This checking can be performed as text is entered and can also be invoked by the user at any time during the edit session.

The Karel Command Language. While the Karel language is the means by which applications are described, the Karel Command Language, or KCL, is the operational language of a Karel controller. It is used for operation of the controller as well as for on-line program development. Though KCL is an explicit text language, most of the commands can be entered with softkeys so typing is rarely, if ever, necessary. The complete set of KCL commands includes the following categories:

- Program Development. This includes commands to operate the on-line editor and translator and to load programs from secondary storage (including off-line or host storage) to RAM memory.
- Program Control. A host of commands permits the operator to run, abort, pause, and resume programs. System parameters can be used along with these commands to permit single-stepping and breakpointing.
- Data Manipulation. Data can be declared, assigned initial values, saved to secondary storage (including off-line storage), listed, and moved or renamed while in RAM.
- File Manipulation. The customary file manipulation commands are available, including copying, renaming, listing, and directory listing. Commands are also available for device mounting and dismounting (including network devices).
- Miscellaneous. Miscellaneous commands include those needed for diagnostics, utilities, help, status display, and robot calibration.

KCL commands can also be executed from command files, allowing the operator to define an application-dependant power-up sequence and Karel program loading and execution sequences.

Position Teaching. Position data is generally created using the teach pendant. It can also be entered directly via KCL commands at the keyboard. A newly created program is executed by a special RUN-TEACH function. Whenever the new program attempts to execute a MOVE statement to an uninitialized position, execution is suspended and the operator is prompted to teach the position. After the position has been taught, the program execution can be resumed. The programmer is thus automatically led through the teaching phase by the same logic the program executes.

MOTION CONTROL

As previously noted, Karel provides a number of MOVE statements for specifying motion. The type of motion interpolation, speed, and termination conditions can also be controlled. Three basic interpolation methods are supported in Karel. JOINT interpolation coordinates the motion of each robot axis to begin and end at the same time. The overall time of the motion is dictated by the axis that requires the longest time to complete its move. LINEAR interpolation moves the robot TCP in a straight line from its current position to the destination position. The orientation of the tool is smoothly changed to the orientation of the destination position. CIRCULAR interpolation results in the TCP's moving along a circle from initial position to a final position through a via position.

In addition to the interpolation algorithm used to move from position to position, the user can specify the termination conditions for a motion. The four types of termination are NODECEL, NOSETTLE, COARSE, and FINE. Motion termination determines when one motion is completed and the next can begin.

APPLICATION DEVELOPMENT

Karel does not include the large built-in set of special robot options found in many systems, such as palletizing functions, special arc weld functions, and spot weld functions. While these options are sometimes useful, they frequently do not adequately meet the customers' real requirements, with the result that customers either have to alter their applications to fit a special option or they simply do not use it. The Karel language allows easy tailoring of such options by either GMF or the user to suit specific needs, an approach that requires fewer changes to the basic system software and renders it more reliable.

COMMUNICATIONS AND SENSORS

Karel provides an extensive set of communication and sensor functions, including communications based on RS-232 that use protocols such as DDCMP and the General Motors MAP specification. Several options are offered that depend on

the speed of communications, number of communication links needed, and so forth. Basic communication functions include program and data upload and download, data modification, status, and most operator functions supplied by KCL. In addition, the functions necessary for synchronization and control between sensor systems and the robot system are provided. A real-time sensor option provides the means to interface sensor data into the path planning portion of the Karel system and to perform real-time tracking for arc welding, sealing, and other applications.

ACKNOWLEDGEMENTS

The Karel system is the result of the contributions of many individuals. Lothar Rossol, the GMF vice president of research and development, deserves credit for providing the initial impetus to develop Karel. Dr. Berthold Horn, of the Massachusetts Institute of Technology, was a major contributor during the early stages of specification and development. Finally, the entire Karel team deserves credit for the system described here.

REFERENCES

1. Shimano, B. "VAL: A Versatile Robot Programming and Control System." *Proceedings of COMPSAC 79,* November 1979, 878-883.

2. Franklin, J.W., and G.J. Vanderburg. "Programming Vision and Robotics Systems with RAIL." *Proceedings of Robots VI,* 1982, 392-406.

3. Taylor, R.H., P.D. Summers, and J.M. Meyer. "AML: A Manufacturing Language." *The International Journal of Robotics Research,* 1, no. 3, 1982, 19-41.

4. Paul, R. "Wave: A Model-Based Language for Manipulator Control." *The Industrial Robot,* March 1977.

5. Mujtaba, S. and R. Goldman. "AL User's Manual." Stanford Artificial Intelligence Laboratory Memo AIM-323, 1979.

6. "User's Guide to VAL-II." Danbury: Unimation Incorporated, April 1983.

7. Ward, M.R., K.A. Stoddard, and T. Mizuno. "A Robot Programming and Control System which Facilitates Application Development by the Use of High-Level Language." *Proceedings of the 15th International Symposium On Industrial Robots,* September 1985.

A SAMPLE KAREL PROGRAM

```
PROGRAM Sealing2
-- This a simple sealing program.  The program waits for a part
-- ready signal (cycle), moves to a starting position, turns on
-- the sealant dispenser, and seals along the path.  At the
-- completion of the path, the sealant is turned off, the robot
-- moves to a clear position, and the conveyer is cycled.

VAR  part_count, count, timer1  : INTEGER
     start_seam, purge, ready, clear_part: POSITION
     path_1                     : PATH
CONST
  -- Digital Output Signals
       Sealer_on = 1
       Solvent_on = 2
       Air_on = 3
       Cycle_conveyor = 4
  -- Digital Input Signals
       Part_ready = 1
       Sealant = 2

 ROUTINE timeout(signalno:  INTEGER):BOOLEAN
       timer1 = -5000                          -- set timeout to 5 sec
       REPEAT
         IF DIN[signalno] THEN RETURN(FALSE)   -- return FALSE if no timeout
       UNTIL timer = 0
       RETURN(TRUE)                            -- return TRUE if timeout
 END timeout
```

Continued on p. 14

Continued from p. 13

```
----------------Beginning  of  Main  Program  Body----------------------------
  BEGIN
     part_count = 0                                  -- clear part counter
     $speed = 1000                                   -- set default speed
     $motype = joint                                 -- set default motion type
     CONNECT TIMER TO timer1                         -- set up a timer
  WHILE TRUE DO                                      -- set up continuous loop
    FOR count = 1 to 100 DO                          -- set up loop for 100 parts
       MOVE TO ready                                 -- ready position
       Wait_part                                     -- routine call to for part
       WITH $SPEED = 200 MOVE TO start_seam          -- approach start of seam
       Start_Sealer                                  -- routine call to start sealant
       WITH $termtype=NODECEL $motype=LINEAR $speed=600 MOVE ALONG path_1
       DOUT[sealer_on] = FALSE                       -- turn off sealant
       WITH $termtype=NODECEL $speed = 200 MOVE AWAY 20.0
       MOVE TO clear_part                            -- clear part
       part_count = part_count + 1                   -- maintain part counter
       PULSE DOUT[Cycle_conveyor] FOR 200            -- cycle conveyer
    ENDFOR                                           -- end of 100 part loop
    purge_gun                                        -- routine to purge gun
  ENDWHILE                                           --  end of infinite loop
END sealing2                                         --  End of Main Program


 ROUTINE Wait_part
 BEGIN
    IF timeout(part_ready) THEN
       MOVE TO purge
       purge_gun
       WRITE (' Timed Out Waiting for a Part ')-- write error message
       PAUSE                                     -- pause program
    ENDIF
 END  Wait_part

 ROUTINE Start_Sealer
 BEGIN
    DOUT[sealer_on] = TRUE                         -- signal the sealant gun
    IF timeout(sealant)  THEN                      -- wait to see if success
       MOVE AWAY 100.0                             -- if not, move away from seam
       MOVE TO PURGE                               -- move to purge position
       purge_gun
       WRITE ('Timed Out Waiting for Sealant') -- display operator message
       PAUSE                                       -- stop the program
    ENDIF
 END  start_sealer

 ROUTINE Purge_Gun
 BEGIN
      MOVE TO purge                                -- move to purge position
      DOUT[solvent_on] = TRUE                      -- turn on solvent
      Delay 50                                     -- wait 50 msecs
      DOUT[solvent_on] = FALSE                     -- turn off solvent
      DOUT[air_on] = TRUE                          -- turn on air
      DELAY 100                                    -- wait 100 msecs
      DOUT[air_on] = FALSE                         -- turn off air
  END purge_gun
```

ROBOTALK: A NEW LANGUAGE TO CONTROL THE RHINO ROBOT

H.S. Sandhu
Rhino Robots, Inc.
3402 N. Mattis Ave.
Champaign, IL 61821
and
Herbert Schildt
Universal Computing Laboratories, Inc.
PO Box 618
Mahomet, IL 61853

The word "robot" commonly evokes a mental image of the mechanical and electronic components that make up the tangible, visible robotic device. However, it is the invisible component, the software, that defines a robot and distinguishes it from simple hard automation. The first 25 years of industrial robotics were characterized by evolutions in hardware, but since the beginning of this decade the primary focus of robotics research and development has been on software. Sophisticated programming languages for robot teaching and control have been created both as support systems and as substitutes for the traditional teach pendant as a way to train and control a robot. While the pendant is an excellent method of teaching fairly simple tasks such as welding and palletizing, when the job is more complex and external event synchronization must be considered, or when the robot needs to recognize and respond differently to a large number of occurrences, the teach pendant system becomes overburdened.

The solution is a robotic control language for writing programs specifically designed to control a robot. We created RoboTalk especially for the Rhino XR-Series robots. The language is interactive, making it useful both for teaching the robot and for instructing the operator in the principles of programming.

All robotic control languages have certain features in common, such as commands for motion, grip, synchronization, and relative and absolute coordinate adjustments, as well as the control structures common to all standard programming languages. The language must also allow interaction with the teach pendant and its controls. The reason a specific robotic control language is preferable to simply adding robotic control subroutines to a more general purpose language is that there are a number of data structures peculiar to this application area. These include motor position for points in space, the current absolute and relative position of the robot, and the home position used to reference all relative motion. A control language specific to robotics often provides a more natural means of expression than does a more general-purpose computer language.

TRADITIONAL INDUSTRIAL ROBOTIC CONTROL LANGUAGES

Our development of RoboTalk was based on an examination and assessment of several of the more common industrial robot control languages, including Unimation's VAL, Cincinnati Milacron's control language, and Automatix's RAIL.

VAL. VAL uses a BASIC-like syntax with the GOTO being the only form of loop control and the GOSUB used to call subroutines. All VAL's variables are global; truly stand-alone subroutines cannot be created. The teach pendant is integrated into VAL through the editor. As locations are taught, lines of text are added to the program. (This is the method we adopted for RoboTalk.) Once they are taught, the MOVE command can be used to reference these points directly in the program. All spatial locations are either absolute or relative.

One of VAL's best qualities is the ease with which it can be learned by those with programming experience, and even people new to programming can create short, useful programs in a few hours. VAL's major limitation is that it is an old language that lacks modern control structures and does not support stand-alone subroutines.

Cincinnati Milacron's Language. The Cincinnati Milacron language is unique in

that it is essentially based on a directed graph, similar to a road map that has a trip planned on it with arrows indicating direction and circles around points of interest. For a robotic control language, the points of interest are specific operations like close grip or move to a location, and the arrows direct the program flow.

The Cincinnati Milacron language's major advantage is the high level of teach pendant integration that allows the operator to both create a program and guide the robot. Its major limitation is that large programs tend to be hard to comprehend because the program is not in a "normal" computer language format.

RAIL. Automatix's RAIL is the most modern control language since it has Pascal-like control structures. In fact, RAIL looks like UCSD Pascal with robotic control procedures and functions added. RAIL's main asset is that stand-alone subroutines can be created that use local variables. This means, for example, that various programs can share the same weld routine. A minor limitation is the difficulty RAIL presents to someone who has never programmed before. Pascal, and languages like it, are the preferred languages for large, complex programming projects, but are not the easiest languages to learn without adequate software tool environments.

ROBOTALK'S BEGINNINGS

The Rhino XR-Series robots are aimed at three markets: educational, experimental, and hobby. We felt the new control language we were trying to create should address and satisfy all three. Moreover, RoboTalk was to provide more advanced features than the existing languages offered. The following design criteria were agreed upon in October of 1984. RoboTalk would:

- be a general introduction to all industrial robotic control languages;
- be easy to learn and use, even for the novice;
- support an educational environment;
- use all features of both the XR-Series robots, as well as the Mark III controller and the work cell;
- be completely integrated with the teach pendant;
- allow external devices and multiple robots to be synchronized;
- act and feel like an industrial language, with fast response time, smooth execution, and immediate abort capabilities.

Because so many XRs are used in education, we felt this application should be given first priority. We decided to keep the language simple, without sacrificing completeness; to add a trace and debugging facility to help beginning programmers understand the operation of both their program and the robot; and to add a direct execution mode that allows easy experimentation with RoboTalk commands. We decided to write RoboTalk in C for both portability and efficiency. Compiled C code runs nearly as fast as assembly code and is easier to maintain. We also wanted to produce versions for both the Apple IIe and the IBM PC. The Aztec C compiler was chosen because there are versions for both of these computers.

Since RoboTalk was to be hosted by a microcomputer, which, in its default mode of operation is not multitasking and, in the case of the Apple II, had only limited interrupt support, we chose to use polling techniques to control the motors, teach pendant, console, and external devices. Considerable emphasis was therefore placed on the main loop of RoboTalk and the key support routines that drive it. Approximately 70 percent of the development time was spent on the support routines and the main loop. The effort was worthwhile, because RoboTalk acts on a microcomputer the way industrial computers act on their interrupt-driven controllers.

HOW ROBOTALK IS ORGANIZED

RoboTalk is organized into five sections:

- the editor
- the interpreter
- the direct execution mode
- the filer
- the teach pendant support routines

The Editor. The RoboTalk editor, used to create RoboTalk programs, is a cross between the Microsoft BASIC editor found on the IBM PC and the VAL editor. It has three modes of operation: text insertion, command, and teach. In text insertion mode, you enter lines of text that become your program. In command mode you give it editor commands. The editor is line oriented and supports the following text editing commands:

Command	Function
Append	begin inserting text onto the end of the program
Delete	deletes a line of text
Edit	allows modifications to a program line
Insert	puts editor into insert mode from which text can be entered
List	list the program
New	begins a new program
Print	prints the program on the printer
Quit	exits the editor
Skip	go to the specified line

In teach mode, you use a teach pendant to guide the robot through a series of moves that automatically create portions of your program.

The Interpreter. RoboTalk is an interpreted language; the instructions in your program are executed as they are encountered. The part of RoboTalk that actually runs your program is called the interpreter.

Direct Execution Mode. In direct execution mode you can give RoboTalk any command it recognizes and it will be executed directly. You do not need to write a program. This mode is excellent for both experimentation and illustration. Commands can be entered sequentially, allowing a complex series of operations to be explored. To exit this mode, a "Q" is typed and control reverts to the main menu.

The Filer. The filer is used to save and load RoboTalk programs, and to list the diskette's file.

The Teach Pendant Routines. RoboTalk, like VAL, integrates its teach pendant support into the editor. In this way, moves taught with the pendant can be entered into programs. There are three commands that control the teach pendant:

Command	Function
Examine	displays and positions the robot at the specified point and allows changes to the coordinates of that point.
Remove	removes the specified point from the program.
Teach	allows point locations to be taught using the teach pendant.

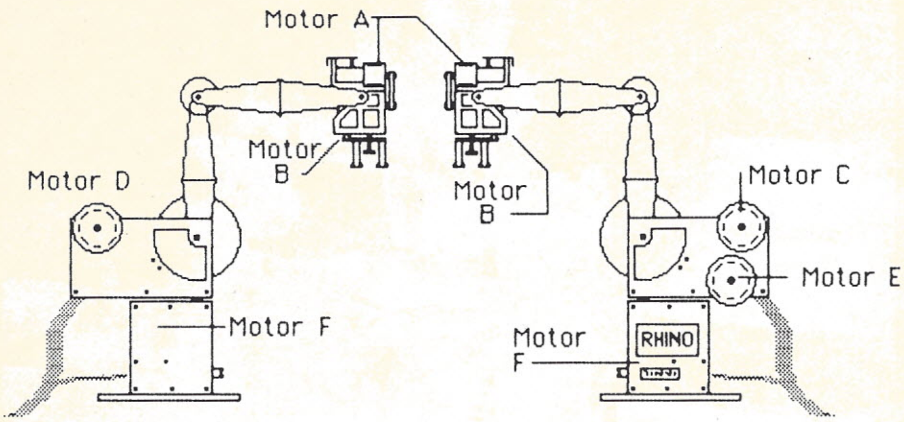


Figure 1. Left and right side views of the Rhino robot indicate the positions of its six motors.

The Rhino's teach pendant allows control of all six of the robot's motors and the two auxiliary motors that run the work cell. The robot's grip acts like a pneumatic hand; it will close on any size object without having to be retaught. The user can, however, instruct the grip to close a specified amount.

Of the teach pendant keys available, only the motor arrow keys and the STORE, END, and HOME keys are used. The others are not needed because RoboTalk automatically supplies these support functions. The arrow keys move each motor forward or backward. END returns control to the host computer. HOME moves the robot to the hardhome position. Each time the STORE key is hit, the current location is remembered and a line of text is generated in the program. For example, if you issued the teach command followed by the point name "BLOCK," and stored this point, you would have the first line of this program text:

```
1    MOVEP BLOCK
2    MOVEP BLOCK1
3    MOVEP BLOCK2
```

Pressing the STORE key two more times creates the second two lines of the program. As a convenience, RoboTalk automatically appends numbers to the end of the point name so that a series of related points can be easily taught and their linkage through a common name remembered.

A SAMPLE ROBOTALK PROGRAM

Let us examine a short RoboTalk program. The diagram in Figure 1 shows the location of each motor on the robot. The simple controller program is listed as follows:

```
1       REM A SAMPLE ROBOTALK PROGRAM
2       CLS
3       TYPE "A PROGRAM TO PICK UP A BLOCK"
4       SETI A=1
5       10 MOVEP BLOCK
6          MOVEP BLOCK1
7          CLOSE -1
8          GOSUB 100
9          MOVEP BLOCK2
10         OPEN -1
11         SETI A=A+1
12         IF A<11 THEN GOTO 10
13      END
14      100 REM WAIT FOR CONVEYOR
15          WAIT 4
16          MOVEGH 100,0
17          RETURN
```

All commands and statements in RoboTalk can be in either upper or lower case. To avoid confusion, we have used upper case.

This program instructs the robot to pick up blocks and place them onto a conveyor belt. It assumes that input signal 4 being set ON signifies that the conveyor is ready and that the next block will wait for the robot to pick it up.

Line 1 is a remark. Remark lines begin with the keyword REM, as is the case with the BASIC command. Everything following the REM to the end of the line is treated as a comment and skipped. Line 2 uses the command CLS to clear the host computer's screen. Line 3 uses the TYPE command to display a message on the host computer's screen. TYPE can also be used to output numeric values. The SETI command in line 4 causes the value 1 to be placed into A, one of RoboTalk's 26 built-in variables. The label "10" begins line 5. All labels must be numbers between 1 and 32,767. Following the label is the MOVEP command. The MOVEP tells RoboTalk to move the robot to the point specified, in this case the point called BLOCK. The MOVEP command is used with points taught using the teach pendant. Line 6 contains another MOVEP instruction. Line 7 is a command to close the robot grip. The argument -1 means to close all the way on any size object. If a positive argument had been given, such as 20, the grip motor would have advanced 20 encoder steps. Line 8 uses the GOSUB command to call a subroutine. Line 9 is another MOVEP command. The OPEN command in line 10 opens the robot grip all the way. The -1 argument is used to completely open the grip, while a positive argument opens the grip a specified number of encoder steps. Line 11 uses SETI to increment the value of variable A. Line 12 shows the IF/THEN statement,

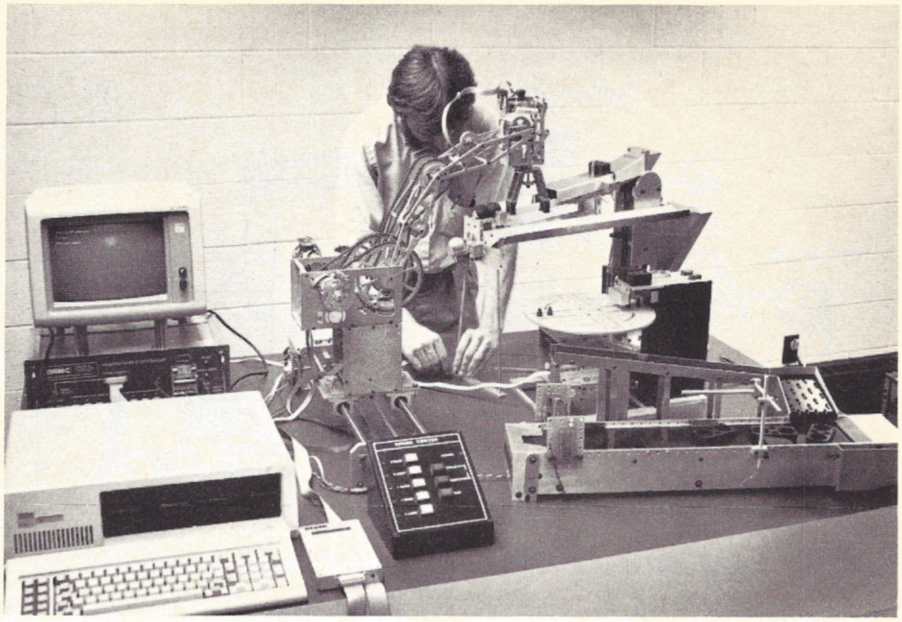


Photo 1. RoboTalk runs on the IBM PC to give the user real-time control over this workcell. The robot can be ordered to put various combinations of balls onto a pallet, and to place the pallet onto a conveyor.

similar to both VAL and BASIC. In essence, it is used here to construct a loop. As long as A is less than 11, the program will loop back to line 5. Line 13 is the END command that signifies the end of the program. It is there to keep the program from falling into subroutines. If there are no subroutines, the command is not necessary. Lines 14 through 17 contain a subroutine that begins with the label 100 and ends with a RETURN, which tells RoboTalk to resume execution at the line after the GOSUB. Line 15 uses the WAIT command to suspend program execution until input signal 4 is set ONE (or TRUE). The MOVEGH command in line 16 advances the G motor by 100 encoder steps. This is an example of direct program control of a motor. All motors on the robot and the work cell can be controlled by either the teach pendant or by explicit program commands such as MOVEGH.

THE TRACE AND DEBUG FACILITIES

Because RoboTalk was intended for use in education, we felt it should have a complete trace facility. When the trace is turned on, the following information is displayed and updated as the program runs:

- position of all motors
- status of input and output ports
- status of the two auxiliary ports
- value of all variables
- currently executing line of source code
- current output

There are two ways to control the trace function. The first is by using the TRON and TROFF commands. These can be placed directly into a program and used to trace specific lines of code. For example:

```
1    MOVE 10,10,200,198,300
2    TRON
3    MOVEP POINT12
4    WAIT -4
5    TROFF
```

will cause RoboTalk to first move the robot to the position specified in line 1, activate the trace screen, execute the next two lines while tracing, and finally turn the trace screen off.

The screen can also be activated dynamically any time a program is running by striking a "T" on the keyboard. It can be turned off by striking the "T" a second time. This allows the user to trace a portion of the program on demand while it is executing.

To aid in program writing, RoboTalk has long error messages and markup, which means syntax and runtime errors can be easily fixed because the line in which the error occurs is displayed on the screen. For example, if the MOVEP instruction were misspelled in line 12, you would see:

```
Syntax error in line 12
NOVEP BLOCK32
   ^
```

RoboTalk treats a stall condition as a runtime error, just as industrial control

languages do. A stall occurs when the robot strikes an object or tries to lift too heavy a weight. The program will stop and the operator can reset the robot using the HOME command, correct the problem, and continue. RoboTalk always knows where the robot is relative to its hardhome position. Even when a stall occurs, all internal motor registers are updated to reflect the actual position of the robot and the program will adjust automatically.

SYNCHRONIZATION WITH EXTERNAL DEVICES

Photo 1 shows a complete workcell under the real-time control of RoboTalk. The operator can use the control box to order various combinations of balls to be placed on a pallet. The WAIT and IFSIG/THEN commands the pallet to be placed onto the conveyor. These commands allow RoboTalk to interact and synchronize the robot with external events. The WAIT command suspends program execution until the specified signal is turned on and the IFSIG/THEN command operates like the IF/THEN except that it uses the status of the input lines as the condition. The OUTSIG command is used to set output signals.

For example, the following program will wait until a car comes down the assembly line by waiting for signal 7, then weld it, and finally set port 3 to signal the completion of the weld:

```
    REM A SIMPLE SYNCHRONIZATION EXAMPLE
    REM FIRST WAIT FOR A CAR
    10 WAIT 7
    REM ONCE 7 IS SET, THEN WELD
    GOSUB 100
    REM SET OUTPUT SIGNAL 3
    OUTSIG 3
    REM SEE IF DONE FOR DAY
    IFSIG 5 THEN END
    GOTO 10
100 REM WELD
```

One major problem with synchronization is the possibility of deadlock, a situation similar to the old comedy routine in which two men at a door each insist that the other go first, with the result that neither gets through. The programmer must use WAIT and IFSIG carefully to avoid this predicament.

Harprit Singh Sandhu is the president of Rhino Robots, Inc. **Herbert Schildt** is the president of Universal Computing Laboratories, Inc.

THE LANGUAGE

The RoboTalk programming language has 28 commands, divided into two groups: program control commands and robot control commands. Program control statements control program flow, while robot control commands run the robot and its controller. The commands are:

Program Control	Robot Control
CLS	AUX
END	CLOSE
GOSUB	HARDHOME
GOTO	HOME
IF	MOVE
IFSIG	MOVE TO
PAUSE	MOVEGH
REM	MOVEGH TO
RETURN	MOVEP
SETI	OFFLINE
TROFF	OFFSET
TRON	ONLINE
TYPE	OPEN
WAITFOR	OUTSIG

RoboTalk has 26 built-in variables—the letters A through Z. It has room for 100 user-taught location points. The IBM PC version can accommodate programs up to 10,000 characters, about 500 lines. The Apple version can handle 2000 characters.

As do all computer languages, RoboTalk combines constants, variables, and operators to form expressions.

- *Constants. RoboTalk has both string and numeric constants. A numeric constant is any integer between −32,768 and 32,767 and a string is a list of characters surrounded by quotation marks. Valid strings include "this is a test" and "this is 100."*
- *Variables. A variable is a storage location in the computer that can hold a numeric value. Unlike a constant, a variable's value can be changed. The command SETI is used to assign values. SETI G=123 gives variable G a value of 123. Each new assignment overwrites the previous value.*
- *Operators. RoboTalk has two types of operators, arithmetic and relational. The arithmetic operators are standard algebraic operations: + (plus), − (minus), * (multiply), and / (divide). These operators are used to form numeric expressions. The relational operators are used in the IF/THEN statement to return a TRUE or FALSE value based on the outcome of the operation. They are: < (less than), > (greater than), and = (equal to). For example, 10<11 is TRUE, whereas 10<9 is FALSE.*

A numeric expression is one or more variable, constant, and operator that follow the standard rules of algebra. For example:

```
10
A+12/5
(B−G) *2
((10−F)+100)−(10−E)
```

Within expressions, all parenthesized groups are evaluated first, then multiplication and division, and finally addition and subtraction. Operators on the same level are evaluated left to right. Parentheses can be used to alter the order of evaluation.

Reader Feedback

To rate this article, circle the appropriate number on the Reader Service card.

72	82	92
Excellent	Good	Fair

ROBOTICS AND INDUSTRIAL REEDUCATION

Barbara K. Lindauer, Ph.D.
Robot Defense Systems, Inc.
471 E. 124th Ave.
Thornton, CO 80241

There are at least three reasons the technologies associated with automation, including robots, will have an impact on the fields of training and instructional technology within this decade:

- a need for skilled technicians
- the development of retraining programs for displaced workers
- training for end users in specialized robotics applications

Robotics is an emerging technology whose effects are only beginning to be felt in the workplace. By next year an estimated 50,000 industrial robots will have been installed [1] with projections of 100,000 to 150,000 by 1990 [2]. Estimates of an annual growth rate for the automation industry are as high as 30 to 49 percent. This rapid expansion has been attributed to the perception that robots, coupled with related types of factory automation, are the means to restore a favorable trade balance and to enhance industrial productivity in traditional manufacturing industries [3]. As with any new technology, there will be a demand for trained professionals, such as those in the engineering and computer science fields, as well as for skilled technicians to install and maintain state-of-the-art equipment. By the end of this decade, an estimated 70,000 jobs will have been created by the robot industry [4], and 5000 to 10,000 skilled technicians will be needed [5].

Automation is being implemented at the fastest rate in the automobile industry, where 25 percent of the robots in the U.S. are in service [1]. Unions, through collective-bargaining agreements, have received extensive company-paid training programs in consideration of economic concessions granted by auto workers. Other industrial unions have echoed the concern that robots are capable of accelerating the displacement of skilled, semiskilled, and craft workers and have included in their labor agreements clauses pertaining to retraining benefits [6,7]. With increasing concern over the potential negative effects of worker displacement due to technological innovation, the AFL-CIO is calling for the creation of a national industrial policy that would include the development of effective employment and training programs [8].

Additional types of automation, long considered the poor relations of industrial robots, will also influence the need for future training endeavors. These specialized robotic applications include such innovative examples as adaptations of arms, grippers, and voice control for the handicapped and elderly; Department of Defense funding for robotics, including the development of autonomous robotic vehicles for defense and physical security; and the creation of affordable, easily programmed robots for educational and personal use.

Technologies intimately related to automation, such as high-fidelity simulation used in computer aided design/computer aided manufacturing sensor systems, and artificial intelligence will support concurrent advances in computer-based, individualized training.

THE NEED FOR SKILLED TECHNICIANS

The original supposition in this discussion was that the increased demand for automation would lead to the creation of jobs for skilled robotics technicians. A recent forecast, based on a survey of management currently involved in the automation field, predicted that 400,000 technicians would be needed to service automated factories by the end of this century [4]. Most of the 10,000 trained technician positions that will become available within this decade will be filled by employees coming from company training programs [5]. However, over 400 high schools, community and technical colleges, and universities are now offering courses in automated systems. How can trainers and educators resolve the need for future technician training without flooding the job market?

Traditionally, technician training in terms of maintenance and installation has been provided by robot vendor companies to clients who have purchased their products. In a recent Office of Technology Assessment survey [9], 93 percent of those companies that produce programmable automation equipment offer instruction to purchasers. The extent of this instruction is limited in scope, with 80 percent of the vendors providing only a single course. Only a third of these companies said they were prepared to provide all the training actually required to operate and maintain

their equipment. Sufficient training was not provided because of cost factors and a diffusion of responsibility to the end users for providing employee education.

The perceived lack of adequate technician training has not been lost on educational institutions, but most of the existing programs offer only limited skills and techniques instruction, rather than training in fundamental disciplines such as mathematics, physical sciences, materials, electronics, and mechanical, fluid, thermal, and optical devices [5]. Piedmont Technical College in Roxboro, North Carolina, has developed an automated manufacturing technology program that teaches operation, installation, maintenance, and repair techniques [10]. The trend for such programs appears to have accelerated nationwide, due in part to the increased attention the media have paid to sophisticated automation technology [11].

To avoid a proliferation of specialists with only currently employable skills, a more broad-based approach with guidelines for a variety of employment situations has been suggested. Scheduled for completion this year is a model curriculum for robotics/automated systems technicians developed by the Center for Occupational Research and Development and the U.S. Department of Education. This new approach to curriculum design includes seven quarters consisting of 80 percent common core subjects (basic and technical skills) and 20 percent specialty core subjects (robotics and sensors). This curriculum was designed to produce graduates with an extensive technical background that will allow them to find jobs in a variety of areas, as well as to provide a foundation for the subsequent retraining necessary to keep up with technological changes [5]. The success of such training endeavors is as now undetermined. One community college–based program reported successful placement of 18 out of 23 of its graduates [12], while another was able to find jobs for only a few of its students [13].

Part of the confusion in assessing industrial needs for technician training lies in separating those trainees who are being hired into the workplace for the first time and who have received formal education in automation disciplines from those workers who have been displaced and who are receiving new skill training in company-sponsored programs. In other words, estimates of the numbers of trained technicians who will be needed, especially for large consumers of automation technology, may be realistic, but at least a portion of these positions will be filled by displaced employees. In-house training to upgrade to necessary skill levels is often mandated by union contracts stipulating that no new technicians are to be hired from outside sources such as community colleges. In fact, 50 percent of the displaced workers are expected to receive training for new positions within the same plant [14]. Another factor limiting the need for large numbers of technicians in the future is expert systems applications in automatic test equipment. As robots become more technologically advanced, they will be able to perform their own routine maintenance, diagnosis, and repair functions [14].

For example, in Michigan alone between 13,500 and 24,000 jobs are expected to be lost to robots during the next five years. At the same time, the robot industry is expected to generate 5000 to 18,000 new jobs [1]. The key issue is that we do not know how many workers in these new jobs will come from the pool of displaced workers or will be graduates of technical training programs. Of the factories using programmable automation, only 22 percent polled in a recent survey sponsored or conducted new technology-related training. Of those that did not offer skill updating, only 18 percent indicated future plans for implementing such programs [9].

In light of this limited data, it appears that a need does exist for vocational/technical/continuing education programs that will provide the automation industry with skilled technicians capable of keeping abreast of advancing technologies [15,9]. Until definite industry trends are discernible, trainers and curriculum developers would be well advised to incorporate robotics courses into general skills categories. This will give graduates a flexible background and allow them to pursue a number of career alternatives.

The Retraining Dilemma. Robots are expected not only to increase productivity and improve product quality, but also to reduce labor costs. Using an example from the auto industry, employee costs including benefits range from $23 to $24 per hour, whereas the cost, including acquisi-

tion and maintenance, for an industrial robot doing the same job averages approximately $6 per hour [16]. According to a recent documentary, the most completely automated factory in the world has reduced an estimated labor force of 2500 to 215 [17]. Estimates vary, but each robot installed in the U.S. is projected as replacing between 1.7 and 6 jobs. By 1990, robots could displace 1.2 million workers [2]. Workers who have lost their jobs because of technological change are expected to reenter the work force through industry- and government-sponsored retraining programs [18].

In terms of designing retraining programs, the critical issue becomes one of calculating the number and types of workers needing to be trained. While some economists and social demographers are predicting higher than normal attrition rates due to automation [19,20], others suggest a minimal effect on the work force between now and the end of the decade [21,22]. What *is* known, however, is that different geographic locations, especially those in the Great Lakes "rust bowl" region, are being hardest hit in terms of plant closings, worker displacement, and transfer of labor-intensive jobs to third world nations [17,2,24]. What educational planners should expect is that somewhere in the neighborhood of 2 to 4 percent of the manufacturing industry labor force will need some form of retraining by the end of this decade [21].

Labor unions have been the main impetus behind retraining program development. The UAW predicts that by 1990, 200,000 of its members, approximately 20 percent of its total membership, will lose their jobs to industrial automation [2]. This time frame coincides with GM's reported plan to install 20,000 new robots [17]. This displacement of workers is expected to occur despite a 15 percent projected growth rate for the American automobile industry. In their 1982 Ford and GM agreements, the UAW obtained provisions for establishing training and retraining programs for current employees, as well as workers who have been laid off as a result of new technologies and new production techniques. As a result of that contract, Ford established the National Development and Training Center to "promote training, retraining, and other skills development opportunities for current and displaced workers" [9]. In August of 1982, the Center began a National Vocational Retraining Assistance plan that provides financial aid (up to $1000 yearly) to laid-off workers who wish to enroll in formal education or retraining programs. The Targeted Vocational Retraining Projects were initiated concurrently to support retraining in specialized skills for occupations where worker shortages existed in specific locations [9]. The "nickel an hour" fund that finances these retraining programs has produced approximately $10 million to support the Ford Center and $40 million for retraining at GM. Job security was also an issue for the most recent collective bargaining agreements. The total value for 1984 contracts is estimated at $1 billion. This amount does not seem excessive since it gives the major auto manufacturers license to employ automation technologies to the greatest possible extent as long as retraining is provided. Similar agreements have been reached by unions in the aerospace, electrical, communications, and steel industries [6,7].

Whereas most of the public's attention has been focused on collective bargaining resolutions to the displaced worker problem, the federal government has in the past funded, under the Comprehensive Employment and Training Act and the Trade Readjustment Assistance Act, programs for retraining workers. Under the newer Job Training Partnership Act, which encourages partnerships for training by public/private sources, approximately $2.8 billion was made available in 1983 to train and employ displaced workers [9]. The efficacy of federal involvement has been questioned, however, since such intervention could be interpreted as a denial of industrial capabilities for sponsoring retraining and, too, the national government might be too far removed from local situations to adequately address training issues [24].

Even without having precise data on the rates of technological change and diffusion, or on the actual numbers of workers being displaced by automation, it is clear that retraining has become big business. With most large-scale retraining efforts just getting underway, we must speculate on the successfulness of retraining endeavors in returning displaced workers to new and productive jobs. In the Office of Technology Assessment's report on automation in the workplace [9], a comprehensive review suggested that retraining programs

benefit most those displaced workers who are young, have more formal education, and have some financial security. Of such individuals, however, only 15 percent participated in retraining programs, ostensibly because of the lack of financial support. In addition, the types of jobs available for displaced workers must be determined. Will all jobs now available or in the future require new, higher-level skills? Or will at least a portion of those displaced find themselves in lower-level jobs and be classified as unskilled?

Reports from two retraining programs in Allegheny County, Pennsylvania, funded by federal, state, and local community colleges indicated that the number of displaced workers dwarfed available resources for retraining. Further, the courses devised were only marginally successful in placing trainees full-time in the areas of their academic specializations. Of 23 students enrolled in a robotics program, 5 dropped out of the course, and five months after the course was finished, 12 were counted as working full- or part-time in course-related fields. In a comparable program aimed at retraining 30 millwrights to become stationary engineers, only 15 had positions 17 months after graduation. The robotics program is being offered again, but the other program has been dropped [25].

If trainers are to be successful in returning displaced workers to full-time occupations, they must develop curricula applicable to a wide range of ages, educational backgrounds, and skill levels. These curricula should include planning for the changes new technologies will make in the work environment. This factor may prove more significant than job requirements for determining worker flexibility [26,27]. Program evaluations must be included as an integral component of whatever nationwide industrial policy is formulated. Existing evaluation data must be viewed as inconclusive because of program differences, environmental differences, problems associated with small samples, and selection bias difficulties [28]. Without the long-range strategic planning implications of such a policy it will be difficult to predict in exactly what skills workers will need retraining or how many jobs will become available. Nor will it be possible to gauge the successes and failures of combined government and privately funded retraining programs.

TRAINING FOR SPECIALIZED ROBOTIC APPLICATIONS

Training needs will not be limited to large-scale retraining efforts or technician training. Small groups of end users will require specialized training as robots are used in a wide variety of nonindustrial categories. The following examples are by no means an exhaustive list of robotic capabilities expected to be developed within this decade, but they can serve as illustrations of new uses of automation technology that require some degree of instruction.

In a joint research project of the Veterans Administration and the Stanford Research Institute, robots with specialized arms and grippers are being programmed to execute simple domestic tasks for people severely disabled by spinal cord injury. Since the project's implementation, end users have been involved in the design, development, and implementation of the project [29]. Not only were they instrumental in the research and development process, but they also provided the necessary material for the creation of training manuals for new end users. As these experimental concepts are expanded to include homebound, physically disabled adults, including the elderly [30], training concepts from the laboratory will be extrapolated to larger, less sophisticated audiences.

The Department of Defense and private industry have recently joined forces to sponsor robotics research and development in the areas of physical security, automatic ammunition loaders, and intelligent maintenance, diagnosis, and repair systems [14]. Autonomous vehicles are viewed as capable of tirelessly patrolling perimeters of secured areas such as nuclear facilities or prisons; the result will be a reduction of personnel needs [31]. Such robots have potential use also in medical evacuation, fighting fires, entering hazardous areas, and defusing explosive materials. These robots will require trained technicians for their maintenance and repair. At issue for trainers are the human factors involved in person-machine interfacing. Training will have to focus not only on technical issues but also on public response to the potential danger of such vehicles and the coordination of person-machine capabilities.

Another area of robotics with potential training impact is the personal robot en-

thusiasts. Personal robots offer relatively unexplored potential for providing domestic help, intrusion detection, and home learning opportunities [32,33,34]. User-friendly training may take on many new meanings within this context. If the personal robot market parallels that of home computers, a whole industry could develop dedicated to instruct relatively unsophisticated end users. Rather than focusing on technical skills, such training would emphasize the interface between human and robot.

CONCLUSION

Whether the issue is new technician training, displaced worker retraining, specialized training for end users, or the translation of automation technology to educational and domestic applications, training will play a key role in this phase of the industrial revolution. Even without definitive strategic planning data, there are already training needs that are not being addressed. The Office of Technology Assessment [9] has recognized four:

- to learn how and by whom technological literacy should be addressed
- to define both long- and short-term instructional systems
- to initiate innovative curriculum design processes
- to identify funding sources for curriculum design and implementation

To these trainer and instructional designer roles the following should be added. First, those involved in training and human factors issues should promote the need for strong industrial planning and program evaluation efforts. It will be difficult to train or retrain employees without a good understanding of the job opportunities of the future. Second, information on the successes and failures of existing large-scale training efforts are essential to the improvement and upgrading of the programs currently being offered. Third, it is imperative to adapt new technologies developed by automation and to apply them to improving instructional media and software design. Finally, as new end-user robotics applications emerge, the training opportunities they represent must not be ignored. An emerging partnership between those interested in human factors within the automation process and those responsible for technological innovation will be established and should experience significant growth through 1990 and beyond.

REFERENCES

1. Hunt, A., and T. Hunt. *Human Resource Implications of Robotics.* Kalamazoo: W.E. Upjohn Institute for Employment Research, 1983.

2. Joint Economic Committee. Hearing on the impact of robotics on employment. March 18, 1983.

3. Hetzner, W.A., L.G. Tornatzky, and K.J. Klein. "Manufacturing Technology in the 1980's: A Survey of Federal Programs and Practices." *Management Science,* 29, 1983, 951-961.

4. Society of Manufacturing Engineers. *Industrial Robots: A Delphi Forecast of Markets and Technology.* Dearborn, MI: Society of Manufacturing Engineers, 1982.

5. "The Course of Robotics Education." *Robotics Today,* August 1984, 40-44.

6. Hanauer, G. "Bargaining for Jobs in the Automated Factory." *California Lawyer,* January, 1984, 29-31.

7. Roberts, M. "A Labor Perspective on Technological Change." Article prepared for the Office of Technology Assessment to be published by the University of Pennsylvania Press.

8. Wachter, M.L., and S.M. Wachter. *Toward a New U.S. Industrial Policy?* Philadelphia: University of Pennsylvania Press, 1981.

9. Office of Technology Assessment. "Automation and the Work Place: Selected Labor, Education, and Training Issues." Washington, DC: Congress of the United States, Office of Technology Assessment, March 1983.

10. "Programming Robotics." *Open Entries.* December 3, 1983.

11. Asfahl, C.R. "Curriculum Dilemmas for Robotics Educators." *Technological Horizons in Education Journal.* August 1984, 98-103.

12. Schreiber, R.R. "Meeting the Demand for Robotics Technicians." *Robotics Today: '82 Annual Edition.* Dearborn, MI: Robotics International, Society of Manufacturing Engineers, 1982. (Article originally published in *Robotics Today,* Summer 1981.)

13. Lee, C. and R. Zemke. "Retraining America: Solutions or Sugar Pills?" *Training.* May 1983, 22-29.

14. National Research Council. *Applications of Robotics and Artificial Intelligence to Reduce Risk and Improve Effectiveness: A Study for the U.S. Army.* Washington, DC: National Academy Press, 1983.

15. Kimbler, D.L. "Engineering Robot Applications and Education." *The Higher Education Journal,* May 1984, 122-126.

16. Foulkes, F.K., and J.L. Hirsch. "People Make Robots Work." *Harvard Business Review,* January-February 1984 (1), 94-101.

17. Cronkite, W. (Narrator). "High Tech: Dream or Nightmare?" (CBS Reports). New York: Columbia Broadcasting Station, 1984.

18. Hollenbeck, K., F.C. Pratzner, and H. Rosen (Eds.). *Displaced Workers: Implications for Educational and Training Institutions.* Columbus, OH: The National Center for Research in Vocational Education, 1984.

19. Bluestone, B. "Industrial Dislocation and Its Implications for Public Policy." In *Displaced Workers: Implications for Educational and Training Institutions.* K. Hollenbeck, F.C. Pratzner, and H. Rosen (Eds.), Columbus, OH: The National Center for Research in Vocational Education, 1984.

20. Bluestone, B., and B. Harrison. *The Deindustrialization of America: Plant Closings, Communi-*

ty Abandonment, and the Dismantling of Basic Industry." New York: Basic Books, 1982.

21. Miller, S. *The Impact of Robotics on the Work Force and Work Place.* Pittsburgh, PA: Department of Engineering and Public Policy, Carnegie-Mellon University, 1981.

22. Parsons, H.M., and G.O. Kearsley. *Human Factors and Robotics: Current Status and Future Prospects.* Alexandria, VA: Human Resources Research Organization, 1981.

23. Smith, D.A., J. Kulik, and E.W. Stromsdorfer. "The Economic Impact of the Downriver Community Conference Economic Readjustment Activity Program: Choosing between Retraining and Job Search Placement Strategies." In *Displaced Workers: Implications for Educational and Training Institutions.* K. Hollenbeck, F.C. Pratzner, and H. Rosen (Eds.), Columbus, OH: The National Center for Research in Vocational Education, 1984.

24. McKenzie, R.B. "Displaced Workers: A Role for the Federal Government." In *Displaced workers: Implications for Educational and Training Institutions.* K. Hollenbeck, F.C. Pratzner, and H. R. Rosen (Eds.), Columbus, OH: The National Center for Research in Vocational Education, 1984.

25. Chute, E. "Job-hungry: Numbers of Dislocated Workers Dwarf Resources for Retraining." *Pittsburgh Press,* August 12, 1984.

26. Argote, L., P.S. Goodman, and D. Schkade. *The Human Side of Robotics: Results from a Prototype Study on How Workers React to a Robot.* Pittsburgh: The Robotics Institute, Carnegie-Mellon University, 1983.

27. Beaumont, R.A., and R.B. Helfgott. *Management, Automation, and People.* New York: Industrial Relations Counselors, 1964.

28. Barnow, B. "The Economic Impact of the Downriver Community Conference: Reactor Comments." In *Displaced Workers: Implications for Educational and Training Institutions.* K. Hollenbeck, F.C. Pratzner, and H. Rosen (Eds.), Columbus, OH: The National Center for Research in Vocational Education, 1984.

29. Awad, R.E., K.G. Engelhardt, and L.J. Leifer. "Development of Training Procedures for an Interactive Voice-controlled Robotic Aid." *Proceedings of the 6th Annual Conference on Rehabilitation Engineering,* San Diego, CA, 1983, 276-278.

30. Engelhardt, K.G., and L.J. Leifer. "The VA RR&D Center." *Generations,* Summer 1984, 38-39.

31. "Birth of the Killer Robots." *Newsweek,* June 25, 1984.

32. Boyer, J. "Robots: The New Helpmates." *Working Woman,* April 1984, 135-136.

33. Watt, P. "Robots Ready." *Infoworld,* May 14, 1984, 12.

34. Watt, P. "Robots Roll onto the Scene." *Infoworld,* June 11, 1984, 52-55.

Dr. Lindauer, who holds a doctorate in psychology, is a research scientist with Robot Defense Systems, Inc.

Reader Feedback

To rate this article, circle the appropriate number on the Reader Service card.

73	83	93
Excellent	Good	Fair

Advertising

* *Correspond directly with company*

Classified Advertising

Personal Robots, Kits and Books at discount prices. **Cal-Robot,** 16200 Ventura Blvd., Suite #223, Encino, CA 91436, (818) 905-0721

MILLER'S WHEEL AND PINION CUTTING. Write To: DAVID G. MILLER, 23½ E. State St., Alliance, OH 44601.

ROBOTICS SUPPLIES: Stepping motors, translator/driver ICs, Hall switches, tools, ICs, transistors, diodes, capacitators, resistors, LEDs, breadboarding supplies, wire, etc. Competitively priced. Quantity pricing on some components so join resources with a friend and save. 20 page, 1985 catalog, $1.00. **Digimation,** P.O. Box 1479, Cullowhee, NC 28723.

RB5X ROBOT, Less than half price, includes sonar, arm, voice/sound package, 16K memory option, charger, asst EPROMs, T-Beat communications software for Commodore C-64, RS-232 interface for C-64, 2 manuals, excellent condition, $2,250 firm. C-64 Commodore, 1541 disk drive, 1526 printer also available. Call 617-356-3563 after 6 pm. or write A. Gaudette, 1 Jay Rd., Ipswich, MA, 01938.

Would like to purchase back issues of Robotics Age magazine for personal collection. Original issue of Summer 1979, Winter 1980, Jan/Feb 1981–May/June 1981. Write to Joseph A. Nowakowski, P.O. Box 29097, Chicago, IL, 60629, U.S.A.

Young Robotics Enthusiast would like to correspond with other enthusiasts involved in robotics, personal, industrial, research & development or just plain collectors of robotic materials. Also would like any news or magazine articles, photographs, posters, brochures, or books concerning robots.. Write to Joseph A. Nowakowski, P.O. Box 29097, Chicago, IL 60629, U.S.A.

USING LOGO FOR ROBOT SIMULATION

B.J. Gleason
71 Bennett Ave.
Kearny, NJ 07032

LOGO is a language that was first introduced by Seymour Papert as part of an experiment in teaching children how to use computers. Turtle robots carrying pens were programmed to move about in patterns that produced drawings. This background might not immediately suggest the language's powerful LISP-based constructs and its usefulness in graphics simulation, but the Atari LOGO cartridge for the Atari 1200XL home computer can be used to simulate a turtle, build it a room, and allow it to wander around running into walls and other turtles. One particularly helpful feature of LOGO is the "demon" that can be used to monitor events, which can then help control robot behavior.

Table 1 lists the events LOGO demons will watch for.

In essence, there are three types of events. The first is a TURTLE-LINE collision. We have four turtles and three pens that leave lines on the screen. When a turtle passes over a line, a demon can run a routine if we have told it to watch out for that event. Next is the TURTLE-TURTLE collision that occurs when one turtle runs into another. Again, if we have told the demon to watch out for that event it will then execute a routine. Last comes SPECIALS. These include the joystick's changing position, the fire button on the joystick's being pressed, and an event timer that is activated once per second.

To instruct the demons to watch for events, we first need to know the event number. These are in the first column of Table 1. For example, when Turtle 0 passes over the line drawn by Pen 0, we see that it is event number 0. Having ascertained the event number, we can use it in either of two ways to test for that event. The first way is the COND statement. The COND # statement returns TRUE or FALSE, depending on whether or not that event number has occurred. We can therefore place it in an IF statement.

```
IF COND 0 [PRINT [TURTLE 0 PASSED OVER LINE 0] ]
```

In the above statement, when the turtle passes over the line the screen will display "TURTLE 0 PASSED OVER LINE 0." The problem with the COND statement is that it only tests immediately. If we are doing other things as well, we might miss the turtle's passing over the line. Fortunately, demons can be used in another way, with a WHEN statement that acts as an interrupt for the system.

```
WHEN 0 [PRINT [TURTLE 0 PASSED OVER LINE 0] ]
```

As soon as an event occurs, we process the routine specified in the WHEN statement, regardless of what else we are doing. This statement is executed only once. We can then go on and process other information, knowing that as soon as that event happens again we will be told that "TURTLE 0 PASSED OVER LINE 0." This is the basis for robot simulation.

TABLE 1
Events and Demons

EVENT NUMBER	TYPE OF EVENT	DEMON
0	TURTLE-LINE	Turtle 0 crosses Pen 0
1	TURTLE-LINE	Turtle 0 crosses Pen 1
2	TURTLE-LINE	Turtle 0 crosses Pen 2
3	SPECIAL	Joystick button pressed
4	TURTLE-LINE	Turtle 1 crosses Pen 0
5	TURTLE-LINE	Turtle 1 crosses Pen 1
6	TURTLE-LINE	Turtle 1 crosses Pen 2
7	SPECIAL	Once a second
8	TURTLE-LINE	Turtle 2 crosses Pen 0
9	TURTLE-LINE	Turtle 2 crosses Pen 1
10	TURTLE-LINE	Turtle 2 crosses Pen 2
11	NOT USED	
12	TURTLE-LINE	Turtle 3 crosses Pen 0
13	TURTLE-LINE	Turtle 3 crosses Pen 1
14	TURTLE-LINE	Turtle 3 crosses Pen 2
15	SPECIAL	Joystick position changed
16	TURTLE-TURTLE	Turtle 3 hits Turtle 0
17	TURTLE-TURTLE	Turtle 3 hits Turtle 1
18	TURTLE-TURTLE	Turtle 3 hits Turtle 2
19	TURTLE-TURTLE	Turtle 0 hits Turtle 1
20	TURTLE-TURTLE	Turtle 0 hits Turtle 2
21	TURTLE-TURTLE	Turtle 1 hits Turtle 2

THE SIMULATION

The robot we are simulating is very simple. It has only one bumper and can detect when it hits a wall but cannot tell the direction of incidence. One of the first programs someone writes for a robot is to have it run around on the floor, bumping into things. We can have our robot do the same by drawing a room on the screen, placing our robot into the room, and letting it go. When it bumps into a wall, we can back it up and head it off in a new direction. This is done in Listing 1.

```
LISTING 1: LOGO LISTING OF SIM1

TO SQUARE :LEN
REPEAT 4 [ FORWARD :LEN RIGHT 90 ]
END

TO DRAW.ROOM
PENUP
SETPOS [-100 -100]
PENDOWN
SQUARE 200
PENUP
SETPOS [ 0 0 ]
END

TO RECOVER
REPEAT 10 [ BACK 1 ]
SETH ( 22.5 * RANDOM 16 )
END

TO SIM1
CS
FS
DRAW.ROOM
WHEN 0 [ RECOVER ]
SETSP 20
END
```

ROUTINE SQUARE. This simply draws a square with each side :LEN units long.

ROUTINE DRAW.ROOM. This sets up our room. We first tell the turtle to lift up its pen so it does not make too many lines. We place it in a lower left corner of the screen (SETPOS [−100 −100]), put its pen down, and tell it to draw a square 200 units on each side. This is a rather simple room, but we could draw a more complicated one if we wished to. We raise the pen again and place the turtle in the center of the room.

ROUTINE RECOVER. This routine is called whenever the turtle hits a wall. We tell it first to back up ten steps, and then to select a new direction. In this routine we use the RANDOM 16 instruction, which will return a value between 0 and 15. We take this number and multiply it by 22.5 so the turtle will turn between 0 and 337.5 degrees. When the routine is finished, the turtle heads off in a new direction. This is the most powerful routine, the one in which we can do the most work. The way we recover from collisions and select a new direction begins to enter the realm of artificial intelligence.

ROUTINE SIM1. This is the main routine, with which we tell the turtle to clear the screen (CS), use the full screen (FS), and draw the room. SIM1 sets up the demons (WHEN 0 [RECOVER]), tells the turtle to begin traveling, and sets its speed (SETSP 20).

As the simulation runs, we might see the turtle bang its head several times before choosing a direction that gets it away from the wall. This is a function of the RECOVER routine and we can, if we wish, begin developing a more intelligent recovery routine for when the turtle hits a wall.

THE SMARTER MACHINE

Perhaps the easiest recovery routine to implement is the LAST-RIGHT method. This is accomplished as follows: When we hit a wall we back up. We know the direction we are traveling, so we look up in a list the *response we used last time to get away from the wall. If the number is less than 0, we have never before encountered a wall while traveling in this direction. So we pick a random direction and travel until we hit something. We then see how far we traveled. If we traveled more than, say, 20 units, then it is a valid direction and we place that value in the list, so we can "remember it" if we hit a wall while we are traveling in that direction again. In time, the turtle will get a good response table built up so that when it hits something it can get away without running into many walls.

I ran statistics on this method, comparing it to that of random choice. With random choice, the turtle picked the proper direction to get away from the walls about 50 percent of the time. With the LAST-RIGHT method, accuracy increased to about 70 percent. With the LAST-RIGHT method, the turtle spends more time exploring and less time banging into walls.

To implement the LAST-RIGHT method, we need to give the robot a way to measure distance. On mobile robot platforms this can be done by a number of methods. With a LOGO simulation we can find out the turtle's location by using the POS command, which returns a list containing the current position. We save this

value, travel to a new position, and then identify the new location. Using the DISTANCE command, we can find out how far we have traveled. We then test to see if this distance is greater than 20. If it is, the direction we picked to get away from the wall was correct and we should save it in the list. If the distance is less, the direction was not a good one and another must be selected.

When I first wrote the routine, I noticed a slight problem. For example, the turtle is traveling at 90 degrees, hits a wall, and picks the direction 270 degrees as a means of getting away. And let's say the best response from the 270 degrees is the 90 degrees. LOCKUP occurs. The turtle will just oscillate back and forth indefinitely. We can prevent lockup by testing for 180 degree reversals. If a 180 degree reversal was chosen, we pick a different direction, as in the above example. Listing 2 is the program that gives this behavior.

LISTING 2: LOGO LISTING OF SIM2

```
TO Q
FS
SETSP 20
END

TO S
SETSP 0
TS
END

TO ITEM :N :OBJECT
IF :N = 1 [OUTPUT FIRST :OBJECT]
OUTPUT ITEM :N-1 BUTFIRST :OBJECT
END

TO FETCH :INDEX
OUTPUT ITEM ( :INDEX + 1 ) :MEM
END

TO GET.RAN
MAKE ''STAT 0
MAKE ''NM RANDOM :MAX
END

TO GET.MEM
MAKE ''MR FETCH :DIR
IF :MR = :MAX [GET.RAN] [MAKE ''NM :MR]
END

TO REPLACE :POS :NEWROW :SH
IF :POS = 1 [OUTPUT SE :NEWROW BUTFIRST :SH]
OUTPUT SE FIRST :SH REPLACE :POS-1 :NEWROW BUTFIRST :SH
END

TO NEW.MOVE
IF :STAT = 1 [GET.MEM] [GET.RAN]
IF :DIR = ( FETCH :NM ) [GET.RAN]
END

TO GOOD.MOVE
MAKE ''STAT 1
MAKE ''MEM REPLACE :DIR + 1 :NM :MEM
MAKE ''DIR :NM
PRINT :MEM
END

TO DISTANCE :POS1 :POS2
MAKE ''X ( FIRST :POS1 )-FIRST :POS2
MAKE ''Y ( LAST :POS2 )-LAST :POS2
OUTPUT SQRT :X * :X + :Y * :Y
END

TO SQUARE :LEN
REPEAT 4 [ FORWARD :LEN RIGHT 90 ]
END

TO RECOVER
SETSP 0
REPEAT 10 [ BACK 1 ]
MAKE ''TRAVEL DISTANCE :OLDPOS POS
IF :TRAVEL > 20 [ GOOD.MOVE ] [ MAKE ''STAT 0 ]
MAKE ''OLDPOS POS
NEW.MOVE
SETH ( ( 360 / :MAX ) * :NM )
SETSP 20
END

TO DRAW.ROOM
PENUP
SETPOS [-100 -100]
PENDOWN
SQUARE 200
PENUP
SETPOS [ 0 0]
END

TO SIM2
CS
FS
DRAW.ROOM
MAKE ''MR 0
MAKE ''NM 0
MAKE ''MAX 16
MAKE ''MEM [16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16]
MAKE ''OLDPOS [0 0]
MAKE ''DIR 0
WHEN 0 [ RECOVER ]
SETSP 20
END
```

ROUTINE FETCH gets the LAST-RIGHT response for the current direction from the list MEM.

ROUTINE ITEM returns the Nth item from a list.

ROUTINE NEW.MOVE tests to see if there is a response in memory. If there is none, a random response is given by GET.RAN. It there is a response in memory, the move is given by GET.MEM. Testing is then conducted for LOCKUP. A positive finding brings a new random response.

ROUTINE REPLACE inserts a new value in a list at position N.

ROUTINE GOOD.MOVE updates memory if the last direction of travel was good.

While this program is running, the turtle's memory, contained in :MEM, is

printed out every time it is updated in the GOOD.MOVE routine. This does not appear on the screen unless we stop the simulation by pressing S followed by a <*RETURN*>. This clears the screen and displays the memory changes. The simulation is restarted by pressing Q<*RETURN*>.

LOGO can use a split screen to simultaneously display both text and graphics. If every FS in the program is replaced with SS, the last three lines of text will remain on the screen, and will scroll as commands are entered. This procedure, however, reduces the size of the room in which the turtle is roaming about. Further, the room must be fully displayed on the graphics portion of the screen or the turtle will jump right over the walls.

Once in a while, a turtle may escape from the room. There are a number of reasons that this happens, but mainly it happens because the system can handle only one demon at a time. The easiest way to get the turtle back into the room without disturbing its memory is to type:

```
ASK # [ SETPOS [ 0 0 ] ]<RETURN>
```

where # is the number of the turtle that escaped. In the SETPOS command, it is assumed that 0,0 is inside the room. If it is not, just change the values to a point inside the room. When the command is executed, the turtle will be transported into the room, still heading in the same direction and at the same speed.

THE NEXT STEP

There are several steps we can do at this point:

1. **Obstacle introduction.** We could install another turtle in the room to act as a randomly moving obstacle for the first turtle. This is simple to do, using Listing 3. We now have to watch for several types of collisions. Turtle 0 can hit the wall, Turtle 1 can hit the wall, and Turtle 0 can hit Turtle 1. We set up the demons and let the turtles roam.

```
LISTING 3: OBSTACLE INTRODUCTION

TO OBSTACLE
ASK 1 [ PENUP SETPOS [0 0] SETSP 15 ]
WHEN 4 [ASK 1 [ BACK 10 SETH RANDOM 360 ]]
WHEN 19 [ ASK 0 [ RECOVER ]]
END
```

2. **Addition of sensors.** We could add more bumpers. Since the turtle knows in which direction it is traveling, we could make the vertical wall from Pen 1 and the horizontal walls from Pen 0. By using another demon, (Demon 1) we could find out if we were hit on the side or the front. For example, if we were traveling 10 degrees and hit Pen 1, which runs north/south, we would know we were hit on the side.

3. **Sonar simulation.** Many robots today carry a Polaroid sonar range finder. We could simulate with another turtle the round trip sound would take from the turtle to the wall and back. To find this distance, we place another turtle where we are and have it move forward until it hits a wall and returns to us. We could even put it in a loop, so that it scans 360 degrees and then gives us the direction of the longest distance. Listing 4 is a sonar routine. With TEST.DIS, we move the sonar turtle forward five steps at a time until it hits a wall. It then outputs the distance from the robot turtle to the sonar turtle.

```
LISTING 4: SONAR SIMULATION

TO SONAR
SETSP 0
TELL 2
PENUP
SETH ASK 0 [HEADING]
SETPOS ASK 0 [POS]
ST
MAKE ''DIS 0
TEST.DIS
HT
TELL 0
SETSP 20
OUTPUT :DIS
END

TO TEST.DIS
IF COND 8 [] [ FORWARD 5 MAKE ''DIS :DIS + 5 ]
END
```

CONCLUSIONS

By using LOGO, we can "build" and test the performance of a sophisticated robot without using any hardware. This method allows us to work out the software before we begin robot construction. Even if the robot is not programmed in LOGO, we can use the algorithms we developed to guide program development in other languages.

B.J. Gleason holds a master's degree in computer science from the New Jersey Institute of Technology.

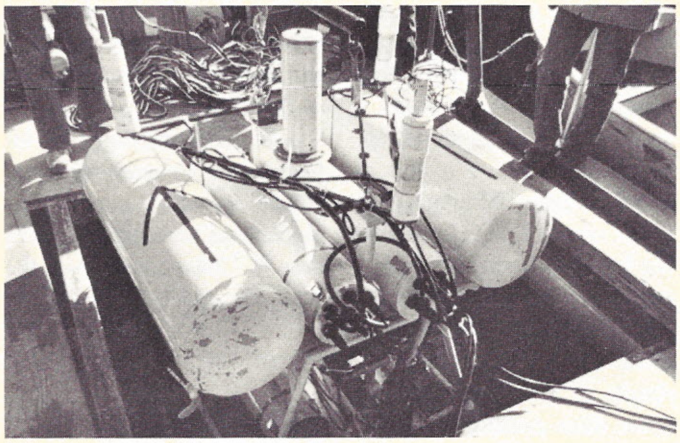
Reader Feedback

To rate this article, circle the appropriate number on the Reader Service card.

74	84	94
Excellent	Good	Fair

Edited by Stephanie vL Henkel

ON THEIR OWN: THE UNTETHERED, UNMANNED, UNDERWATER ROBOTS



EAVE, a prototype untethered, unmanned, submersible vehicle, was built at the University of New Hampshire's Marine Systems Engineering Laboratory.

Public interest in robotic submarines was recently awakened by a tethered, unmanned, underwater vessel's search and retrieval mission. Bell Telephone Laboratories at Holmdel, New Jersey developed the Scarab (Submersible Craft for Assisting Repair and Burial) to dig up, repair, and rebury undersea cables. The sub's normal diving range is 6000 feet, but when an Air India flight went down off the coast of Ireland this past June, the Scarab descended some 6700 feet to find and raise the plane's voice recorder. With its sonar-sensitive "ear," the robot tracked down the source of the voice recorder's pings. A television "eye" sent information up the tether cable to the crew on the host ship. An AT&T Communications spokesman said the sub could probably reach 9000 feet without mishap.

The Scarab is tethered to the mother ship by a 10,000-foot cable that sends down electrical power and commands and returns television signals and data from the sub's on-board minicomputer. On the host vessel, one crew member controls the Scarab's propulsion system, another mans the robot's television cameras and manipulators, and a third monitors the minicomputer.

Less in the public eye, but very much of interest to the scientific community is a robotics research program being conducted at the Marine Systems Engineering Laboratory of the University of New Hampshire in Durham. For the past five years, an annual workshop has been held on unmanned, *un*tethered, underwater vehicles. The first conference attracted 20 participants; this year's June gathering drew 200 conferees from Canada, France, Italy, Japan, and Sweden, as well as from the United States.

> ***Your systems have to be a lot more reliable because there's no one there to take care of them. These are high-anxiety-level vehicles.***

Dick Blidberg, an electrical engineer and associate director of the lab, described untethered undersea robots as "a technology still in its infancy." Robotic submarines, he said, "offer a lot of flexibility in mechanical design since with no human crew, life support systems are not necessary and you don't have to consider safety factors." Further, getting rid of the tether eliminates problems caused by fouled cables that can make it impossible to recapture the vessel. On the other hand, new problems are raised by the absence of crew and tether. "Your systems have to be a lot more reliable," Blidberg said, "because there's no one there to take care of them. These are high-anxiety-level vehicles."

Blidberg and his colleagues have built EAVE (Experimental Autonomous Vehicle), a prototype maintenance and exploration robot that will eventually be able to clean up underwater structures and survey valleys 37,000 feet below the ocean surface. Among the project's sponsors are the university, the Minerals Management Service division of the U.S. Geological Survey, the National Science Foundation, the Office of Naval Research, and various industrial grants.

The prototype is a 4 by 4 foot cube of open space frame design. "It looks, if you stretch your imagination a bit, like the Starship Enterprise," Blidberg said. It is at present "hydrodynamically inefficient," in Blidberg's words, but the early stages of development are a time for free-wheeling experimentation and discovery. "Once it works on paper," Blidberg said, "we want to get it into the water to see if it works there." (The university maintains a lab on Diamond Island in New Hampshire's Lake Winnepesaukee for actual underwater testing.)

With neither human crew nor tethering cable to give it orders, how is the robot sub to find its way around, and collect and transmit data? The answer is logical but not at all simple: by means of a knowledge-based guidance and control system. Three acoustic transponders will be sunk in the area to be surveyed, each tuned to a particular frequency. The robot, emitting acoustic pulses at 120 KHz, will listen with its sensors to the echoed response from the transponders and navigate by triangulation. For some ap-

plications, a single beacon will suffice, Blidberg said.

The prototype robot has an on-board computer with 128 Kbytes of memory, but the final version will have 4 or 5 Mbytes. A UNOS operating system made by Charles River Data Systems of Framingham, Massachusetts guides the craft around obstacles in its path. One problem, Blidberg said, is running symbolic languages in a real-time environment. For instance, acoustic data from the robot's depth sounder determines the slope of the ocean bed. If the floor is rising too fast, that information is transmitted as a symbol. The Prolog interpreter mounted on the operating system translates symbols into commands to carry out course corrections as they become necessary.

The operating robotic submarine will collect video data during its runs and send it back streamlined. By removing all redundant data from the video frame, Blidberg said, and transmitting only pertinent data, the information can be condensed into 2 or 3 frames per second.

In addition to exploratory applications, the robot will be used to clean underwater structures. Stashed in the vessel will be a load of lithium salts that respond to being mixed with sea water by producing a high-pressure gas. The gas emerges in the form of bubbles that scour away mineral deposits, a process akin to sandblasting. The robot will use a pair of arms, "huggers," to attach itself to the object being cleaned so the force of the bubbles won't propel it violently backward.

In Blidberg's estimation, some of the most rewarding results of his work have not come from creating a brand-new widget: "We get very excited about finding new ways of tying the new technology to ocean science." ■

MARKET RESEARCH

Computer aided design and engineering, computer- or programmable controller-controlled equipment, and automated parts tracking will show large gains within the next five years, according to a survey recently published by the **National Electrical Manufacturers Association** (NEMA). Both large and small firms share in the trend to automate, and systems integration is high on the list of factory automation projects. Flexible manufacturing cells and systems are expected to enjoy a four-fold growth, and local area networks will increase three-fold.

The impetus to automate comes from within the manufacturer's organization, the report continues, with the strongest advocates being engineering, senior general management, and corporate management.

An earlier NEMA study identified the lack of in-house expertise in software as a major bottleneck in automation. The new study indicates help is not seen as forthcoming from computer and software vendors, who ranked last on a list of nine potential sources of information. Producers of manufacturing equipment were rated as first, with trade press, trade shows, seminars, professional associations, trade associations, independent consultants, computer manufacturers, and software suppliers following in descending order.

A market survey conducted by **Venture Development Corporation** (VDC) revealed that integrated plant management and control is the most important trend in process control since the 1970 introduction of distributed control.

U.S. consumption of integrated plant management and control systems is expected to increase at an average annual rate of 13.4 percent over the next five years. According to the VDC survey, microcomputer-based systems, including personal computers, offer the highest growth potential, with an annual growth rate foreseen as 18.1 percent. Programmable controllers are increasingly important to the process industries, which will contribute to an anticipated growth rate of 16.6 percent in shipments of the controllers.

The second half of this decade is expected to see an increasing percentage of total capital expenditures allocated to plant automation equipment and services. Plant communications networks and plant management software will show the largest increases.

PEOPLE

► **Michael N. Forino** is the new president of **United States Robots.** He was most recently president and CEO of Hubotics, and has also been in upper management with International Robomation and General Automation.

► **Imaging Technology Inc.** has announced three promotions: Clyde W. Sylvia as vice president for finance and chief financial officer; Patricia Mostika as director of human resources; and Ronald A. Massa as vice president for sales. All three have previously worked for Imaging Technology.

► **Robert Turner** has been promoted by **LASR Robotics, Inc.** to U.S. district sales manager. He has held several previous positions within LASR.

► **Microbot, Inc.**® has appointed **Dino Papoutsis** product manager for educational and developmental robots. He has previously worked for Luxtron Corp., Bio-Rad Laboratories, American Cyanamid, and Diagnostic Corp. of America.

► **James N. Carro** has joined **Ridge Computers** as vice president of marketing and sales. He had been with General Electric Co. for the past 20 years, most recently with that company's Calma Division.

► **William D. Fletcher** has been appointed vice president of special projects by **Allen-Bradley**. He had been head of the company's Drives Division since 1980, and will be succeeded in that position by **Robert L. Swift**, who was

formerly vice president and general manager of the Packaged Control Products Division. Swift will be succeeded by **Terry Nelson**, who was that division's director of operations.

► **Colby Computer** has announced the election of **Richard P. Ettinger, Jr.** to the company's board of directors. Ettinger was one of the original founders and chairman of the board of Wadsworth Publishing Co., Inc. He is chairman of the board of trustees of the Educational Foundation of America.

► **Lester J. Colbert** has joined the board of directors of **Adaptive Intelligence Corp.** Colbert is CEO, chairman of the board, president, and director of Xidex Corp. Prior to joining Xidex, he was with Reichhold Chemicals, Inc. as vice president, board member, and member of the corporate executive committee.

CORPORATE NEWS

► **International Robomation / Intelligence** has announced a 300 percent increase in revenues to $2.4 million, making it the first vision company to report profits. IRI introduced its first vision system in 1982 and has since produced 400 systems for industrial customers. Profits for the third fiscal quarter of 1985 were $31,000.

► **American Technologies Inc.** has closed three new financing arrangements to promote further growth in its factory automation business. One of the arrangements is a $2 million private placement with the Massachusetts Mutual Life Insurance Company. Funds from the note will provide additional working capital for the company's Automated Factory Systems Division, a supplier of robotic and vision systems.

► **WMI Robot Systems, Inc.**, a wholly owned subsidiary of **EDS Technologies, Inc.**, has changed its name to **EDS Technologies, Inc.–Robotics Division.** The new division has also announced an exclusive agreement with **Horyu Control Engineering Co. Ltd.** of Japan to sell and service the Japanese firm's line of robot controllers in the U.S., Canada, and South America.

► **Parker Hannifin Corp.** has signed an agreement to acquire **Schrader Bellows Inc.**, a subsidiary of Scovill, Inc., for approximately $77.5 million in cash. Schrader Bellows, a leader in pneumatic automation products, will augment Parker's products as part of its Fluidpower activities, according to Parker president and CEO Paul G. Schloemer.

► **VSI Automation Assembly** has announced its new status as a wholly owned subsidiary of the Japanese company **Nitto Seiko Co. Ltd.** VSI had been the exclusive North American distributor for Nitto Seiko equipment modules and automated assembly systems, including SCARA robots, end-of-arm tooling, and pick-and-place mechanisms.

► **Tactile Robot Systems** and **AnaPraxix, Inc.** have formed a joint venture to develop advanced software for robotic control. The software will be used in conjunction with tactile sensors to identify, orient, and place objects in materials handling and assembly operations.

► **Gould Inc.** has announced that it intends to reduce its semiconductor operations, taking an approximate $250 million special write-off in the 1985 second fiscal quarter. The company cited continued deterioration of the semiconductor market and resulting losses in its semiconductor operations. The write-off will be used to address obsolete assets, plant reductions, inventory valuations, offshore facilities, and interest in overseas joint ventures.

► **Teledyne CAE** has ordered 11 Maestro systems from **Manufacturing Data Exchange** (MDX). The Maestro product, a DNC/Factory Feedback System, will integrate 20 separate NC machines with a Digital Equipment Corp. VAX-11/780.

► **Microbot, Inc.** has announced its intention to enter the literary field with a series of new texts on applied robotics. The four proposed books will be part of a turnkey curriculum that will be used in conjunction with the Microbot TeachMover and the Alpha II industrial robots.

New Products

Compact Vacuum Gripper

Barry Wright Corp. is offering a vacuum gripper with built-in, adjustable contact and grip sense. The VGC100 gripper weighs 4.5 oz. and has a lift capacity of up to 30 lbs. Cup sizes are available in diameters of ½ in. to 4 in. An optional vacuum transducer mounts on the gripper base and allows operation on standard shop air.

The VG102 vacuum gripper system provides axial compliance and permits the gripper to adapt to objects randomly oriented up to 60 degrees from the centerline. It has a lift capacity of 3 lbs. and cup sizes of 5/8 in. to 1/4 in. diameter.

For more information, contact: Peter N. Cholakis, Product Line Manager, Barry Wright Corp., 700 Pleasant St., Watertown, MA 02172, telephone (617) 924-2929.

Circle 30

muLISP-85

Soft Warehouse, Inc. is offering a new version of muLISP that provides the personal computer user with a window-based artificial intelligence programming environment. muLISP runs on any generic MS-DOS computer and is said to execute three times faster than its competitors and to store two to three times as much program and data.

muLISP has 260 primitives, the use of up to 512 Kbytes, exact and approximate rational arithmetic in addition to integer arithmetic, and user-definable macros as well as functions. Soon to be released are a native-code and a pseudo-code compiler.

For more information, contact: Soft Warehouse, Inc., PO Box 11174, Honolulu, HI 96828-0174, telephone (808) 734-5801.

Circle 32

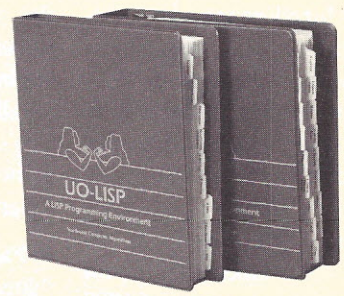
Industrial Environment for a Teaching Robot

Prep Incorporated and Eshed Robotec, Ltd. have created an integrated industrial environment for their Scorbot teaching robot. New additions include the teach pendant, indexing rotary table, linear conveyor, and 12 volt DC servomotors.

The 30-key ER III teach pendant enables the operator to: teach the robot up to 100 positions in space; write programs of up to 250 lines using Scorbase software; transfer programs written from the pendant to a host computer; call up programs from the host computer to the pendant; find the relative and hardhome positions; run a demo program stored in the pendant's memory; and perform emergency braking. It can be adapted to operate on other industrial robot systems.

For more information, contact: Director of Marketing, Prep Incorporated, 1007 Whitehead Road Ext., Trenton, NJ 08638, telephone (609) 882-2668.

Circle 31



UO-LISP

UO-LISP, a powerful implementation of LISP on microcomputers, is available on the IBM PC and compatibles in a package from Northwest Computer Algorithms. The system is geared for professional programmers, engineers, researchers, and educators who want mainframe functions from their personal computers.

The kernel of the UO-LISP interpreter contains over 200 standard LISP functions; precompiled library packages expand the system to over 400. The CP/M-based system comes with a 300-page reference manual and over 30 packages, including those associated with compiling, debugging, program development, extended arithmetic, document processing, a higher-level language, translator writing, and editing.

For more information, contact: Northwest Computer Algorithms, PO Box 90995, Long Beach, CA 90809, telephone (213) 426-1893.

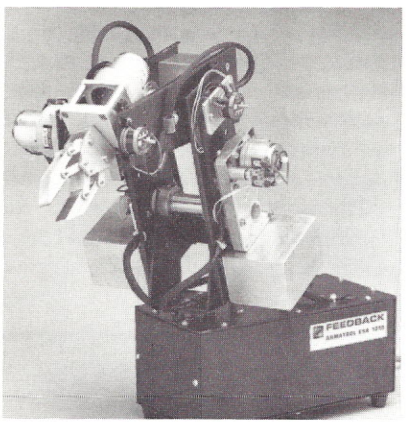
Circle 33

New Products

Enhanced Terrapin LOGO

Terrapin, Inc. is offering Enhanced LOGO version 3.0 for use with the Apple II microcomputers. LOGO 3.0 contains two copies of LOGO on one disk, one for 64K and the other for 128K machines. It requires only 7 seconds to load. In addition, users can access the editor and retain the image they were drawing. Terrapin will update versions 2.0 and 1.3 for its earlier customers.

For more information, contact: Lorraine Chouinard, Director of Marketing, Terrapin, Inc., 222 Third St., Cambridge, MA 02142, telephone (617) 492-8816. Circle 34



Versatile Teaching Robot

The Armatrol ESA 1010, available from Feedback, Inc., is designed for instruction on the university, vocational-technical, military, and industrial levels. The robot features independent waist, shoulder, elbow, and wrist axes, a separate motor-driven gripper, continuous position feedback on all axes, and comes complete with processor and parallel link gripper. The gripper can open up to 2 in. and exerts a force of up to 2 lbs. Reach is 11.5 in.

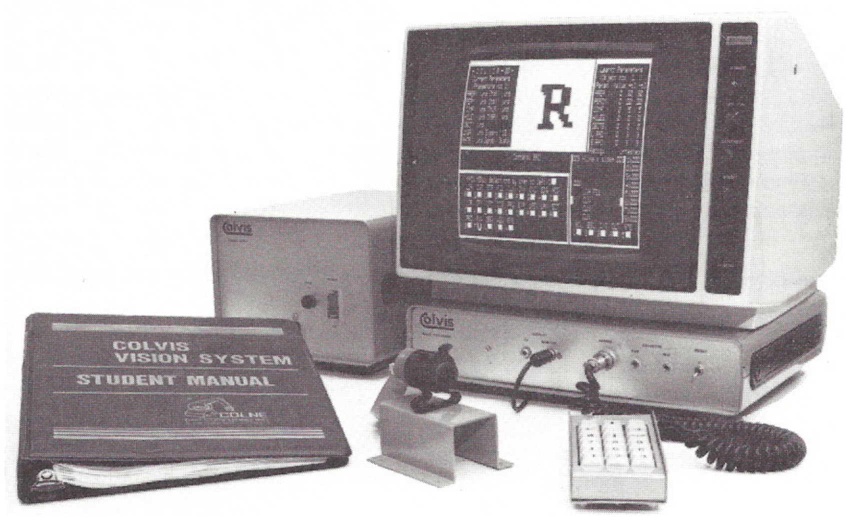
The educational package that comes with the robot includes an interface and control board, cable, integral power supply, and the Armsoft #2 instructional software and manual. The robot can be controlled by Apple, Commodore, Timex 1000, and AIM65 computers.

For more information, contact: Feedback, Inc., 620 Springfield Ave., Berkeley Heights, NJ 07922, telephone (201) 464-5181 or (800) 526-8783 (outside NJ). Circle 35

Vision System for Education

The Colvis Vision System from D&M Computing, Inc. is a self-contained package consisting of a compact, 32 by 32 pixel, solid-state sensor camera with stand, connected to a dedicated microcomputer; a keypad; a 12 in. CRT; a power supply unit; and a student manual. The Z80-based computer system has an RS-232 interface for use in stand-alone mode or in conjunction with a standard 8-bit micro. The system operates at high speed, features a scrolling program display, and is fully programmable.

For more information, contact: D&M Computing, Inc., PO Box 2102, Fargo, ND 58107, telephone (701) 235-7743 or (800) 362-3145, ext. 117 (outside ND). Circle 36

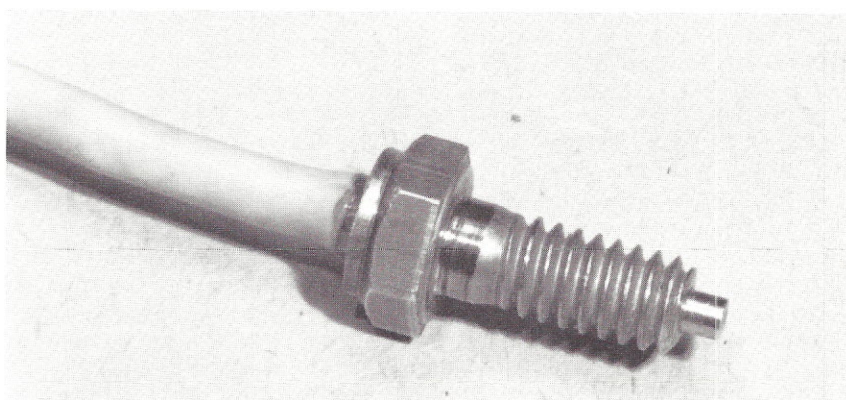


Tactile Sensor for Robots

A set-point touch sensor for the robotics and material handling industries has been introduced by Tactile Robotic Systems. The sensor can be used to determine object presence and position, or to control insertion force and gripper force. It can also be used in weight measurement applications.

Standard units are available for maximum force from 4 oz. to 10 lbs. with 0.050 in. travel and field adjustable set-point. The unit is packaged in a 1/4–20 threaded body. Outputs are relay output or logic level voltage.

For more information, contact: James B. Hawkins, Tactile Robotic Systems, 1172-D Aster Ave., Sunnyvale, CA 94086, telephone (408) 241-1520. Circle 37



New Products

Microcomputer Version of ANSYS

ANSYS-PC/LINEAR, a subset of ANSYS, a software package for structural, thermal, fluid, electrical, and static electromagnetic analyses, is available from Swanson Analysis Systems, Inc. for the IBM PC/AT and PC/XT.

The element library contains two- and three-dimensional spars, beams, solids, shells, springs, and masses in addition to a three-dimensional general matrix. Full-color graphics and animation are standard features. Plotting capabilities include single-window displays of hidden and nonhidden line plots. Detailed on-line documentation is provided. An allowable linear static wavefront of 288 degrees of freedom permits complex analyses.

ANSYS-PC/LINEAR is said to solve a variety of structural problems—determining displacements, forces, stresses, strains, natural frequencies, and mode shapes. The hardware required is: an IBM PC/XT with an 8087 processor or an IBM PC/AT with an 80287 processor, a minimum of 512 Kbytes of memory, a 10-Mbyte disk on the XT or a 20-Mbyte disk on the AT, a standard IBM color graphics monitor, and a parallel printer port. IBM compatibles with similar configurations are also acceptable.

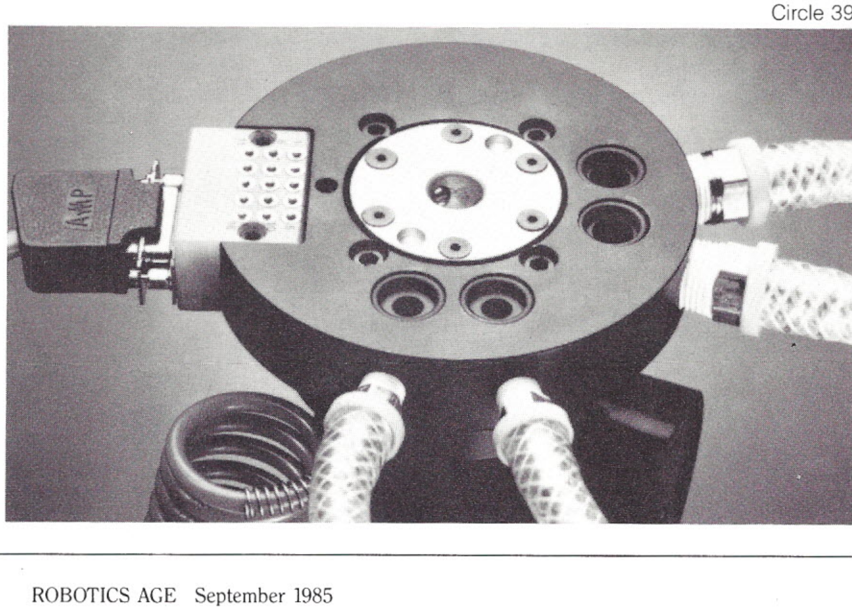
For more information, contact: Suzanne C. Batt, Manager, Marketing and Training, Swanson Analysis Systems, Inc., Johnson Rd., PO Box 65, Houston, PA 15342-0065, telephone (412) 746-3304. Circle 38

Robot Hand Exchange System

XChange Horizon™ is an advanced version of Applied Robotics, Inc.'s original XChange system that incorporates two coaxial cable connectors for video cameras in addition to the 31 electrical and four pneumatic connectors. The system also has a one million cycle warranty. The new coaxial capability allows vision systems (including stereovision) to be accessed by the robot arm on an as-needed basis. A single robot equipped with XChange Horizon can handle all the tools used in an assembly operation and then simply exchange the tools for visual inspection of parts orientation.

Another advantage of the system, according to the company, is that it allows serial communications to be passed through the robot/tool interface, creating "smart tools" that perform specific tasks. The coaxial coupling allows data (such as the complex signal patterns associated with test probes) to be protected by the shielding of the coaxial cabling while being transmitted long distances through environments prone to electromagnetic interference.

For more information, contact: Charles T. Harris, Marketing Manager, Applied Robotics, Inc., 18 Avis Dr., Latham, NY 12110, telephone (518) 783-1800. Circle 39



SCARA Robot for the Classroom

Robotmart is marketing a SCARA type robot developed by Reekie Research Co., Ltd. of England and aimed at teaching students the principles of industrial SCARAs. The Mini-SCARA™ has four degrees of freedom. Model A is a DC stepper motor version with pneumatic Z stroke at the end of the arm, and Model B is all-electric with a stepper motor-driven Z-motion. Both models offer interchangeable end effectors. Tool tip speed is up to 20 in./sec.; repeatability varies by speed and workload.

Programming is done on the IBM PC or other microcomputer connected to the robot by a cable from the parallel printer port to the Mini-SCARA's electronics. The electronics can control three robots simultaneously, operate robotic peripherals, or control optional material handling devices to simulate factory automation.

For more information, contact: Jack Heald, Robotmart, PO Box 1071, 4140 Oak Circle, Boca Raton, FL 33431, telephone (305) 394-3723. Circle 40

Classroom Training Computer

The CSA Micro 68000 from Computer System Associates is a self-contained teaching computer designed to train engineers and technicians by demonstrating logic analyzers and breadboard prototyping. The package includes a 6 amp switching power supply, full hexadecimal keyboard, two RS-232 serial I/O ports, 32-bit parallel I/O, 68000 Versabus computer board, Pete Bug keyboard monitor, 16K RAM and 32K ROM, and a complete set of manuals.

For more information, contact: Patricia Chouinard, Computer System Associates, 7564 Trade St., San Diego, CA 92121, telephone (619) 566-3911 or 274-7391. Circle 41